Design, Implementation and Testing of a Solution to Measure Quality of Life using Mobile Devices

Pedro Miguel Loureiro Mourão
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
FCUP-DCC
2017

Orientador
Miguel Coimbra FCUP-DCC

Coorientador
Pedro Gomes IS4Health

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

Measuring the Quality of Life (QoL) on patients of cancer is very important to help doctors to understand the state of the patient in their daily lives, as well as to simply increase their quality of life. The solution we propose is a mobile application to measure the quality of life on patients of cancer, through questionnaires that can be answered in the comfort of their homes. Every application has been developed in modules to be easily adaptable to various types of questionnaires, scenarios and different systems. In addition, we use a multiplatform development approach to make application available on the two major operating systems today: iOS and Android. The application is being tested in the Instituto Português de Oncologia do Porto (IPO) starting July 2017, in a group of patients focing the following pathologies: Esophageal Neoplasia and Esophagogastric Neoplasia. The patients in this specific IPO group have to answer to a questionnaire about their quality of life through our application in the first, third and sixth month of treatment. The questionnaires used in these tests are: Quality of Life QUestionnaire Esophageal 18 (QLQ-OES18), Quality of Life Questionnaire Esophagogastric 25 (QLQ-OG25) and Quality of Life Questionnaire Cancer 30 (QLQ-C30). There are no results yet, but we hope they will tell us whether the patient undergoing treatment improves their quality of life or not. Analyzing the results and finding out if they are actually able to answer the questionnaire themselves at home is also a very important result for our work. Therefore, at the end of the tests we hope that most patients will be able to have the questionnaire answered at home and that the results have improved throughout the treatment.

Resumo

Medir qualidade de vida num paciente de cancro é muito importante para ajudar os médicos a perceberem o verdadeiro estado do paciente no seu dia-a-dia, como também para simplesmente melhorar a qualidade de vida. A solução que propomos é uma aplicação para dispositivos móveis para monitorizar a qualidade de vida dos doentes de cancro, através de questionários que podem ser respondidos no conforto de suas casas. Toda a aplicação foi desenvolvida em módulos para que seja facilmente adaptável a vários tipos de questionários, cenários e diferentes sistemas. Além disso, usamos uma abordagem de desenvolvimento multiplataforma, para disponibilizarmos aplicação nos dois grandes sistemas operativos atuais: iOS e Android. A aplicação começou a ser testada em Julho de 2017 no Instituto Português de Oncologia do Porto (IPO), num grupo de pacientes com as seguintes patologias: Neoplasia Esofágica e Neoplasia Esofagogástrica. Os pacientes deste grupo específico do IPO têm que responder a um questionário sobre a sua qualidade de vida, através da nossa aplicação, no primeiro, terceiro e sexto mês de tratamentos. Os questionários usados nestes testes são: QLQ-OES18, QLQ-OG25 e QLQ-C30. Ainda não existem resultados mas esperamos que estes nos indiquem se o paciente ao longo do tratamento que é submetido melhora ou não a sua qualidade de vida. Analisar os resultados e percebermos se estes de facto conseguem ou não responder por eles próprios ao questionário em suas casas também é um resultado muito importante para o nosso trabalho. Por isso, no final dos testes esperamos que a maioria dos pacientes consigam ter respondido ao questionário em casa e que os resultados tenham melhorado ao longo do tratamento.

Contents

List of Tables

List of Figures

Listings

Acronyms

API Application Programming Interface

CSS Cascading Style Sheets

DOM Document Object Model

EORTC European Organization for their Research and Treatment of Cancer

EuroQol European Quality of Life

EQ-5D EuroQol Five Dimensions Questionnaire

EQ-5D-3L EuroQol Five Dimensions Questionnaire with Three-Level Scale

EQ-5D-Y EuroQol Five Dimensions Questionnaire Young Version

EQ-5D-5L EuroQol Five Dimensions Questionnaire with Five-Level SCale

EQ VAS EQ Visual Analogue Scale

FACT-G Functional Assessment of Cancer Therapy-General

GPS Global positioning system

HRQoL Health-Related Quality of Life

HTML HyperText Markup Language

IS4H Interactive Systems for Healthcare

IPO Instituto Português de Oncologia do Porto

JSON JavaScript Object Notation

JSX JavaScript with a syntax extension

MSAS-SF The memorial symptom assessment scale short form

MVC Model-view-controller

NDK Native Development Kit

QoL Quality of Life

QLQ-C30 Quality of Life Questionnaire Cancer 30

QLQ-OES18 Quality of Life QUestionnaire Esophageal 18

QLQ-OG25 Quality of Life Questionnaire Esophagogastric 25

RSCL Rotterdam Symptom Checklist

SDK Software Development Kit

SF-12 Short Form 12

SF-36 Short Form 36

SIP Sickness Impact Profile

VAS Visual Analogue Scale

VDOM Virtual Document Object Model

WHO World Health Organization

XML eXtensible Markup Language

Chapter 1

Introduction

The treatment of chronic diseases like cancer is one of the biggest challenges in medicine. Every year, around the globe, there are thousands of deaths caused by these diseases [1]. Therefore every measure should be taken to improve the quality of life of cancer patients since the quality of life is important to measure the impact of a chronic disease [2]. Being able to monitor the quality of life of patients with cancer remotely, helps hospitals and health centers to reduce costs [3]. Quality of life is a hard to define concept, since it must contain both objective and subjective indicators, approaching the patient's personal life. For example, a physiological dimension gives important information to the doctors, even though these aren't very important to the patients [2]. So it's very important to be able to draw social and psychological dimensions to really understand the patient's condition [4]. Aside from these two dimensions, the patient's physical health, well-being, development and activity are also factors to take into account [4]. The questionnaires are able to measure all previously spoken dimensions. In addition, when we use the questionnaires as a means of measuring quality of life, we allow patients to answer to the questionnaire from their homes using mobile devices.

All this motivated us to design, implement and test a quality of life monitoring solution using mobile devices.

The questionnaires used in our solution are validated by the European Organization for their Research and Treatment of Cancer (EORTC) - Quality of Life Questionnaire Cancer 30 (QLQ-C30), Quality of Life Questionnaire Esophagogastric 25 (QLQ-OG25) and Quality of Life QUestionnaire Esophageal 18 (QLQ-OES18) (see attachment A, B and C) - and we choose according to the group of patients where we are testing the solution. The group of the Instituto Português de Oncologia do Porto (IPO) where we are going to test the solution are patients with the following pathologies: Esophageal or Esophagogastric Neoplasm. In the first case, patients have to answer to the questionnaire QLQ-OES18 plus the QLQ-C30, while in the second case they have to answer to the QLQ-OG25 plus QLQ-C30 questionnaire.

Our solution is aimed to be used by the patients themselves, and not someone previously qualified or trained (eg. nurse). Thus, it is expected that answers will be more honest and, at

the same time, with fewer constraints to the patients. It is possible for the patient to answer to the questionnaire anywhere at any time. In this way, patients are not questioned about their quality of life at a less appropriate time, when they are emotionally and physically affected (eg. accumulated fatigue). However, we fully understand that the aggravated state of some patients, or their physical or emotional state, is a hindrance to answering the questionnaire alone. Therefore, we do not rule out the possibility that in some cases it may be justified that a or trained person administering the questionnaire.

1.1 Institutions

All the work presented in this thesis was made from the Interactive Systems for Healthcare (IS4H) spin off in IPO.

IS4H

IS4H is a technology-driven SME that delivers interactive systems for healthcare by integrating different existing technologies to promote personalized healthcare delivery, efficient tools for health care professionals and educational tools.

The developed products and services are the result of a close relation with several research labs and healthcare institutions on Portugal, Europe and Brasil.

Being their main focus on the development of digital auscultation systems. It has developed an interactive patient-simulation platform (IS4Learning) that allows teaching, examination and practicing of auscultation.

IPO

The IPO, E.P.E. mission is to provide healthcare services, in due time, patient focused, not overlooking prevention, research, training and education in the area of Oncology, with the purpose of assuring high levels of quality, humanism and efficiency.

It has the vision that with minimum treatment times and maximum healing ratios, the community shall view the oncological patient as a chronic patient, without stigmas and with quality of life. Technology serving the patient.

Their values represent the commitment that the decisions made by IPO-Porto are on the best interest of the people it serves and employs. Thus, the IPO-Porto structure of values includes five vectors:

- **Quality:** as a "Sistema Nacional de Saúde" (National Health System) hospital, the priority of IPO-Porto is assuring high quality and clinical safety services for the locals. IPO shall learn from the people who use the clinical services, from the Institution's human resources, from the best practices in SNS and beyond. Quality shall be guaranteed by

developing the labour force, strengthening it so it can provide high quality service, safe, effective, and focused on the patient.

- **Integrity :** IPO shall treat its users with dignity and respect, promoting equity, appreciating diversity and offering high healthcare standards. The decisions shall be honest and responsible, in the best interest of the community it serves. IPO-Porto shall promote a labour force which acts in an open way and with integrity. It shall constantly search for ways to build services around the users' needs, meeting at the same time the employees' needs and expectations.

- **People:** people are the core of all the services in SNS: the people who IPO-Porto serves, the people it employs, the people who become members and managers of the Institution, the people who make it possible for the system to be funded and the people who make the system as a whole. The professionals are the hospital's most valuable resources. The Institute shall support the employee so that he may establish an individual contract with each patient, user, co-worker, manager and partner.

- **Excellence:** IPO-Porto wants people, from different geographical or professional areas, to trust the services provided in the Institution. The Institute shall view the future and plan its services based on the needs of the local community, hoping for better continuous improvements in the results of treatments and care it promotes. Is shall use the best available scientific evidence to provide effective services, whether in terms of clinical results, use of financial resources, or others.

- **Community:** IPO-Porto recognizes it is more than a hospital care provider; it is a resources consumer, a waste creator, trusting in the local network of transports. The Institute is a big employer, so it is a significant partner in the local economy. It shall reflect the community's responsibilities in the decisions it makes, working in partnership with the community, volunteers and other organizations to effectively contribute for the local people's lives and reduce its environmental impact.

1.2 Features of our solution

Our solution need available in multiple platforms. Therefore it was very important to study the best approach to develop the solution. All the user interface must be simple, intuitive and have great usability. Since our audience is composed of ill people in complex treatments, the development of the solution gains a lot more responsibility. It's common that during an exhausting day, the patient might lack the emotional availability to answer a questionnaire about their quality of life. Our solution wants to give more comfort to the patient by letting him answer in his own home whenever he feels calm and available. The features that distinguish a solution are:

- Availability of solution on Android and iOS.

- Development of a solution that can be adaptable other contexts/pathologies.
- Allows selfreport at anytime, anywere.
- User authentication.
- Communication with a central server (data transmission).
- Notifications to warn users to answer the questionnaire.

1.3 Potential problems

As we already told, the fact that out audience is made of ill people in complex treatments some potential problems may arise, such as bad data or even the lack of it. In addition, doctors' acceptance of the results obtained may not be the best. For example, if doctors look at the results and don't find them relevant, the application becomes irrelevant.

1.4 Document's structure

Chapter 2 contains important information about quality of life and a small study about the questionnaires doctors use to measure the quality of life of patients giving more focus on cancer patients. Chapter 3 contains our technological approach such as both the native and multiple platform development. We focused on the latter because we are going to use the React Native framework from Facebook. In chapter 4 we talked about the most important features of React Native framework. At the end of the chapter we listed all the libraries we are going to use in our solution. In Chapter 5 we present the complete detailed process of application development with workflows and software architecture. Chapter 6 is where we present the results and the analysis of the results obtained in the preliminary usability test. Finally, we present the conclusions of our work in chapter 7.

Chapter 2

Quality of Life

Defining Quality of Life (QoL) isn't easy, since different definitions exist in different studies. Cancer is a disease that kills a lot of people every year and, in addiction, puts patients in a complicated situation. These people face hard treatments where sometimes results are seen as more important that their quality of life. In our solution we choose to use questionnaires in order to measure quality of life because it's simple and we can promote self-administration. Questionnaires are divided in two groups: general and specific. Both have the same goal: monitoring and measuring quality of life, either if the person is in treatment, or not. The questionnaires we have used in our solution are validated by the European Organization for their Research and Treatment of Cancer (EORTC). These questionnaires were chosen according to the group of patients that we will be able to test our solution.

2.1 What is quality of life?

According to the World Health Organization (WHO), quality of life is a state of complete physical, mental and social well-being, and not simply the lack of disease. In other words quality of life is an evaluation of an individual's well-being. However there are other accepted definitions seeing as quality of life may have different meanings for different people, as well as for different professional areas [5][6][7]. For example, according to *Ben-Chien* [8], there are many quality of life meanings as there are people in the world since everyone may have a different opinion on what is important in their lives. On the other hand, *Baker E et al.* [9] says that there are as many meanings as people who study quality of life. When speaking about disease, clinical trials or treatments the concern is on the evaluation of the aspects that are affected by disease or treatment. And this is how the term Health-Related Quality of Life (HRQoL) was created. This includes the procedure in which the well-being of an individual is measured [10][6]. It's impossible to separate the disease from the personal and social context of the individual [11]. The measures used to assess the patient's quality of life are an important part of the planning and evaluation of the clinical treatment [11]. Furthermore they guarantee the evaluation to focus on the patient and not on the disease [11]. These measures may vary in each study or in each evaluation, however

they generally focus on aspects such as health, physical and toxicity symptoms, emotional and cognitive behavior, and social well-being [10]. Because of this, all collected information may help both the patients and the doctors make the best decisions. It is believed that this information helps patients deal with their disease and treatment [10][11].

2.1.1 Quality of life in a cancer patient

Every year a lot of people die all over the world because of cancer [1]. For example, in the year 2014, 287.000 people died in Hungary due to this chronic disease. In the same year, Portugal counted 197.000 deaths. Finland, the country with the least casualties, counted 173.000 deaths. The cancer treatment's priority should be controlling the tumor [12]. That's why diagnostic and treatment have a huge impact on a patient's quality of life [12]. Remote monitoring devices allow the patients to be controlled from a distance, enabling them to have more autonomy. Real time monitoring has a major role in chronic disease treatment. This approach allows a patient's quality of life to be evaluated, comfortably, from their own houses allowing them to be evaluated at any time they want. *Feld* [13] refers that the quality of life assessment in clinical trials makes a lot of sense, since there are several points (emotional, social, physical, among others) which can affect the patient's daily life. Also *Morris et al.* [14] believes and emphasizes that assessment of quality of life before and during cancer treatment is very important, since it's important to monitor several metrics that can influence the quality of life of every patient.

2.2 How do we measure quality of life?

Medicine and technology are connected. For example, in large hospital, screenings can be done electronically and automatically [15] [16]. It is possible to use technology to improve a patient's quality of life through intelligent environments. The sensors are already widely used in medicine [17]. These are used as medical instruments in the hospital, clinics and even at home. They provide patients and physicians with very important information, such as information and perceptions of physiological and physical health states, diagnostics, etc. According to *Hanson et al.* [18] there are three major types of sensors: physiological sensors, biochemical sensors and environmental sensors. The information collected and monitored by the sensors can be extracted through signal processing [18]. With the technological advancement, wireless sensors appeared [17][19]. They can monitor the patient's vital signs without any use of wires. This upgrade removed some constraints of a wired sensors, like the lack of mobility.

2.2.1 Questionnaires

Another way to monitor and evaluate quality of life is via questionnaires.

Questions such as "How is your quality of life?" are considered metrics for the evaluation

of quality of life. However, this type of question provides very limited information [2], and so they are made up of a set of grouped questions divided into small sub-sets. There are two large groups of questionnaires for quality of life measurement: the general questionnaires and the specific questionnaires. The first type, as the name implies, is intended for general use, illness or condition. These questionnaires can also be applied to healthy people. Among the wide range of general questionnaires we have Sickness Impact Profile (SIP), Short Form 36 (SF-36) and EuroQol Five Dimensions Questionnaire (EQ-5D). Specific questionnaires, on the other hand, are used for certain diseases or to a certain end, for example for the aftermath of a treatment. Some examples are Quality of Life Questionnaire Cancer 30 (QLQ-C30), Quality of Life QUestionnaire Esophageal 18 (QLQ-OES18), Quality of Life Questionnaire Esophagogastric 25 (QLQ-OG25), Functional Assessment of Cancer Therapy-General (FACT-G) or the Rotterdam Symptom Checklist (RSCL), all of which are cancer-specific. Table 2.1 shows a brief summary of some of the most popular questionnaires.

Table 2.1: Synthesis of some study questionnaires.

Questionnaire	Type	Number of questions	Time (around)
SIP	General	136	20-30 minutes
SF-36	General	36	<SIP
EQ-5D	General	s/a	2 minutes
FACT-G	Specific	27	s/a
QLQ-C30	Specific	30	s/a
QLQ-OES18	Specific	18	s/a
QLQ-OG25	Specific	25	s/a
RSCL	Specific	30	s/a

SIP

SIP was designed primarily to evaluate new treatments and the health levels of a population. It is divided into 12 categories, addressing physical issues and functional aspects. However, there is no global question about quality of life or health. The questions are essentially focused on aspects such as daily activities or what the respondent can and does not do. The questions in the questionnaire are written in the negative form, for example instead of asking how the respondent thinks he is understood, it states "I am hardly understood ", being at the discretion of those who fill out whether or not [20].

SF-36

SF-36 assesses a person's overall health status and is designed to address a large market failure: very long questionnaires, which take a long time to complete. This questionnaire was designed to evaluate general health concepts, not to be a specific questionnaire for a special group age, disease or treatment. This questionnaire is about physical, social and emotional functioning. It is divided into 8 distinct concepts: physical well-being, functional well-being, pain, health in general, vitality, social function, emotions and mental health. In addition to these 8 concepts, there are still global gender issues: "Compared to a year ago, how would you assess your overall

health now?". There are also questions aimed at testing the physical limits of the survey, for example if one could walk more than one and a half kilometer or lift a designed object x times [20].

There is a simpler version of this questionnaire, the Short Form 12 (SF-12), that only contains, as its name implies, 12 questions.

EQ-5D

The main objective of the EQ-5D is to evaluate the physical, mental and social functioning of the respondent with a rather simple form. It is divided into 5 groups: mobility, self-esteem / self-care, daily activities, discomfort and anxiety. Each question on the EQ-5D has three possible answers.

The main question of the questionnaire is represented by Visual Analogue Scale (VAS) 20cm vertical, where the respondent will have to represent his current health status on a scale of 0 to 100. The zero represents the "worst state imaginable" and one hundred represents the "best state imaginable".

FACT-G

FACT-G is a cancer-specific questionnaire. Its version 4 is organized in physical welfare, social / family welfare, emotional well-being and functional well-being. As it happens with SIP, some FACT-G questions are posed in the negative form. Individual questions are formulated in the first person, contrary to what happens with the QLQ-C30.

QLQ-C30

Designed for self-management since it was developed based on simplicity and easy filling. It can be applied in several contexts, but it is applied in clinical trials of cancer treatments. This questionnaire has the particularity to address a set of individual questions that assess additional symptoms reported by patients, such as loss of appetite, insomnia, colds and diarrhea [20]. In the version 3.0, many questions, have four distinct levels of response from the "not at all" to the "a lot" [20]. QLQ-C30 is available in several languages and second [20] it is sensitive to patient's differences.

QLQ-OES18

With this questionnaire from the same family as the previous one, the only considerable differences are: only 18 questions instead of 30, and usually more directed, as the name implies, to esophageal cancer patients. [21]

QLQ-OG25

As with the previous questionnaire, the relevant differences are the 25 questions that constitute the questionnaire and the fact that it is more directed to patients with esophageal gastric cancer. [21]

RSCL

According to [20] RSCL has been widely used in European cancer trials, covering very similar areas to QLQ-C30. However, there are some peculiarities which distinguish it: introductory text in the questions and include some questions of daily activities. This second characteristic focuses not on whether the respondent makes X or Y in their daily life, but rather whether it was able to execute X or Y if it was requested. For example, the respondent may not go to but it could answer whether or not he was able to go, if requested [20]. RSCL has 30 questions, with four possible response scales: "nothing," "a little," "a little more," or "a lot" [20].

2.2.2 Questionnaires used in our solution

The questionnaires we use are: QLQ-C30, QLQ-OES18 and QLQ-OG25 (see attachment A, B and C) . They are all modules of the EORTC to evaluate the quality of life in cancer patients. As we have already mentioned, the assessment of health-related quality of life is increasingly important, as it can benefit clinical research and patient care, particularly cancer patients [21].

The QLQ-C30 questionnaire was designed to be easy to complete [20] and covers 5 functional scales: physical, role, cognitive, emotional, and social [20]. In addition, it includes three scales of symptoms: fatigue, pain and nausea and vomiting, and general questions about patient health [20]. This questionnaire is also concerned with secondary symptoms such as dyspnoea, loss of appetite, insomnia, colds or diarrhea [20]. According to [20] the QLQ-C30 was designed to be a modular instrument where the central part of the questionnaire intends to evaluate general aspects of quality of life, although the remaining more specific questions about the cancer in question are very important to be able to evaluate the state in fact of the patient.

In 1986 a study was done with the QLQ-C30 questionnaire, where it was administered to 305 patients with lung cancer [22]. The average time taken by a patient to respond to the questionnaire was 11 minutes and, for the most part, they did not need any kind of help or assistance [22]. The results of the study were quite satisfactory [22], concluding that the QLQ-C30 was a very reliable and valid questionnaire to evaluate the quality of life of cancer patients.

There are several studies in many countries where good patient acceptance results for the QLQ-C30 questionnaire are presented, for example in [23] [24]. Many of the studies refer to the status of patients when the questionnaire was first administered and then the second time after the treatment they underwent. Often the state of health has improved.

According to *Lagergren et. al.* in article [25], QLQ-OG25 is highly recommended for completing the QLQ-C30 questionnaire when evaluating quality of life in patients with esophageal or gastric cancer. A study with the objective of evaluating the Mexican-Spanish version of the QLQ-OG25 questionnaire was presented at [24] with very positive results. The QLQ-OES18 questionnaire was also tested with a population of Taiwan at [26] and again the results were quite satisfactory, concluding that it is possible to use the Taiwan version of the questionnaire to assess the quality of life of esophageal cancer patients.

The Portuguese version of QLQ-OES18 and QLQ-OG25 questionnaires were adapted by [21] and they were tested in 30 patients. After this study, the two questionnaires began to be used in clinical contexts and for scientific purposes [21].

2.3 Who measures quality of life?

Questionnaires to monitor and evaluate patients' quality of life can be self-administered, as well as administered by suitably qualified and trained individuals. It is possible to see a brief summary of the advantages and disadvantages of the modes of administration in Table 2.2 [2]. According to *Fayers et al.* [20], several studies show that independent evaluations, conducted by other professionals, relatives or people close to the patients, differ from the responses obtained when they are confronted with the responses of the patients to the questionnaires. These differences are justified because when another person evaluates the patient, interpretation errors may arise. In some cases, the person who is doing the evaluation may make an excessive evaluation, while in other cases may neglect certain symptoms or complaints of the patient [20]. For example, a cancer patient, when exposed to chemotherapy treatments, his medical staff tends to ignore and accept as natural symptoms such as nausea and vomiting [20]. As we described in Chapter 1, one of the important factors of our solution was to give patients freedom to be able to respond to the questionnaire when they felt comfortable and not in a specific setup. In addition, *Loring et al.* [27] stresses the importance of self-management for chronic diseases patients. The same article further reports as well that the patients studied had minor improvements in their fewer trips to medical services. For all these reasons, our preference fell into self-management.

Table 2.2: Synthesis of advantages and disadvantages of administration mode

Administration modes	Advantages	Disadvantages
Administered by a qualified and trained person	• Guarantees conformity • Decreases errors • Maximizes rates responses	• Requires resources • Possible patient constraints • Limits • Scheduling issues
Self-administration	• Less expensive • Less constraints • Can answer whenever he wants	• Possible misunderstandings • Possible increase of questions that get forgotten/not answered

2.4 Monitor and evaluate quality of life using smartphones

Mobile devices can and should be used to complement health services. In addition, they can also be used by both doctors/medical students and the patients for learning purposes. According to *Robinson et al.* [28], simplicity, organization and instant communication are some of the characteristics that smartphones offer. In 2013, A small survey, that showed interesting results, was conducted among students at the University of Birmingham, UK [28]. About 32% of the college students answered the questionnaire. From those 32%, about 59% had a smartphone. And those students who both answered the questionnaire and owned a smartphone, 37% used, or at least tried to use, their smartphone to help their learning process. Another study reported by *Bielli et al.* [29] and highlighted by [30] [31] consisted of sending structured questionnaires to 97 cancer patients where they had to answer 10 questions from their mobile device. The results of this study were: 56 patients, corresponding to 58% of the population tested, answered the questionnaire. Yet, the 42% who didn't want to participate in the study reported difficulties in using the mobile phone, and perhaps that is why they refused to complete the questionnaire. This study was done with a 10 question questionnaire, extracted from the The memorial symptom assessment scale short form (MSAS-SF), which addresses pain, lack of energy, worry, weight loss, cough, difficulty sleeping, shortness of breath, trouble urinating, lack of appetite, and difficulty to concentrate. The percentage of patients who did not respond to the questionnaire corresponds to an older age group.

2.4.1 Applications for mobile devices related to health

Individuals with chronic diseases are particularly concerned about their diet, as is the case, for example, of patients with obesity or diabetes. Thus, it is important to accurately monitor all their meals. A possible strategy for monitoring ingested food is the image analysis tools for identifying and quantifying food that is consumed in a meal, as presented by *Zhu et al.* in [32].

This way, mobile devices are essential for collecting and processing all information. However, this strategy can be problematic, since it is necessary that the mobile devices have a high image resolution, a strong memory capacity and good network connectivity [32]. Patient reporting is very important so that doctors can adjust their treatments to the needs of each patient [33]. MHealth or mobile health (defined as the practice of medicine supported by mobile devices [34]) makes data collection (by the patients and subsequent sharing) feasible and quite simple. That data is sent to the doctor in real time. Health and physical activity are linked [35]. A fully automated physical activity tool was presented by *Hurling et al.* in [35]. The testing period of this tool took a total of 9 weeks. During the first 4 weeks, about 85% of the participants started a weekly session in the online portal, while in the remaining weeks the adherence was 75%. The most accessed information on the portal was activity charts, exercise lists, and a small instant messaging chat. This tool uses the Internet, email, and mobile devices. Teledermatology, through mobile devices, is an innovative technique for the screening of skin cancer that aims to increase

patients access to dermatological care [36]. A test was performed on the use of teledermatology to track cancer through imaging using mobile devices. According to *Lamel et al.* [36] the results obtained in this test were quite good, since the degree of agreement between the diagnoses given by the specialists and the tool was 81

Chapter 3

Technology

When we are developing a mobile application, in order to keep up with the market, it is important that this same application is available on several platforms. Yet, developing a mobile application specifically for each operating system, the so called native development, in addition to being a very time consuming process, is a process that involves large costs. To overcome this problem frameworks cross-platform (introduced in the section 3.2) have been developed and evolved along the years. There are several frameworks for multiple platform development, which will be presented in section 3.2, referring our choice to the React Native created and maintained by Facebook, as can be seen in section 3.3.

3.1 Native Development

Developing a native application is the same as saying that we are developing an application to run on a particular processor and with a specific set of instructions [37]. Therefore, while developing an Android application, that same application will only run on Android. In the case of native Android development, Android Studio [1] is commonly used, since it provides all the necessary tools (debugging, performance tools, flexible compilation, among others) to native Android programming. It is still possible to use the Native Development Kit (NDK)[2].This native development kit is a powerful toolkit that allows the usage of programming languages like C or C ++ with Android. In addition, the NDK allows access to physical components of the device, such as touch input. This kit is usually used for above-average performance, reducing latency, or simply reusing other programmers' C or C ++ code.In the native development for iOS, the iOS 10 Software Development Kit (SDK) [3] is used, where it is included new Application Programming Interface (API)s and resources. We can none several advantages to develop a native application, among them the technological compatibility, the offline operation (with the browser cache), the speed of its performance, the user interface (hide browser buttons, navigation gestures, specific

[1]Android Studio: https://developer.android.com/studio/index.html
[2]NDK: https://developer.android.com/ndk/index.html
[3]IOS SDK: https://developer.apple.com/ios/ IOS

interfaces for the operating system in question) and security [37]. However, there are also some disadvantages such as a higher cost of development and the non-possibility to reuse code for other platforms, as opposed to multiple platform development [37].

3.2 Multiple platform Development

Within multiple platform there are different ways to develop mobile applications, each with its advantages and disadvantages. In the next four subsections, we will present mobile web application development, hybrid application development, cross platform development, and application development using an interpreter, summarizing and concluding our choice for React Native in Section 3.3 of this chapter.

3.2.1 Web Development

A mobile web application is simply a web application adapted to the mobile format. As it runs through a browser, as long as there is a browser installed, it is possible to access the application, regardless of the operating system (cross-platform). The techniques used to develop a mobile web application are equal to web development, such as HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. [38][39][40][41][37]

3.2.2 Hybrid Development

Hybrid development is a type of development where the application has access to native components (native APIs), but the development is similar to the development of a web application - HTML, CSS and JavaScript. Therefore, it is possible to claim that hybrid development is an intermediary between web development and native development. A hybrid application is displayed in a Web browser, embedded within the application itself, called webview [4]. When running within a webview , a hybrid application suffers from some limitations, such as not being able to access operating system notifications or simply not being able to run the application in the background. In contrast, the development of hybrid applications is more affordable, entails lower costs and cross-platform code sharing is possible. Although the application has access to native components, this capability is also limited since calling the native APIs requires a layer of hardware abstraction. You can only access the camera, the Global positioning system (GPS), and other device hardware components. [38][39][40][41] Among the hybrid frameworks we have Ionic [5], Cordova [6] and Adobe Phone Gap [7].

Ionic

[4]Webview: native component that encapsulates a browser
[5]Ionic: http://ionicframework.com
[6]Cordova: http://cordova.apache.org
[7]Adobe Phone Gap: https://build.phonegap.com

Ionic is a hybrid framework for the development of mobile applications, through JavaScript, CSS and HTML. Applications built by Ionic are a webview, yet the application binary is native. You can access native platform components, however there are limitations, for example you can't access system notifications. According to Ionic's own official website, iOS performance is higher than Android's.

Cordova

The Cordova can be installed together with Ionic or individually. As it happens with Ionic, an application built with this framework runs on top of a webview, being possible to access some native components of the platform, mainly physical components (camera for example).

Adobe Phone Gap

Applications are built using HTML, JavaScript and CSS. However, there is very little support for native components. It is possible to use this framework alone or combined with other frameworks (eg. JQuery or Sencha) to be able to produce better interfaces, as explained in [42].

3.2.3 Cross Platform Development

Cross-platform development, better known as cross compiler, is a compiler capable of producing executable code, that is, native code even though it is written in a non-native language. This type of development can't reuse or share code at 100%, since it will always be necessary native code of the platform in question [38]. Cross-platform development is dependent on cross-compiler efficiency [41]. An example framework that uses this approach is Xamarin [8].

Xamarin

It is possible to develop applications using this framework in C, Java and C#. There is access to native interfaces as well as native APIs, that is, they have access to native performances according to the framework's official website itself.

3.2.4 Development by means of an interpreter (native scripting)

Native scripting is achieved through the use of an interpreter. This interpreter is the responsible for executing the code during runtime, so it is possible to access native components through native APIs using JavaScript. The great advantages of this type of development within multiple platform development is that it is possible to use and call native APIs and the possibility of reusing code between platforms. In addition to being able to use and call APIs, it is possible to perform the tasks that a natively developed application performs, since the communication is done through the interpreter. However, performance is assumed to be lower when compared to native applications. [38][41][39]

[8] Xamarin: https://www.xamarin.com

Example of frameworks in this category are the Appcelerator Titanium [9], NativeScript [10], and React Native [11]. They all work based on JavaScript.

Appcelerator Titanium

It is an intermediate level between JavaScript and native services (natively translating JavaScript code). The end result of application development is a fully native application.

NativeScript

An application built using this framework is compiled even for native, having a runtime close to the native runtime, according to the official website itself. It uses proprietary markup languages and supports widgets.

React Native

An application built using this framework is compiled even for native, having a runtime close to the native runtime, according to the official website itself. It uses proprietary markup languages and supports widgets.

3.3 Conclusions

In this section we will justify our choices. Let's start by justifying why we chose multiplatform development (3.3.1), then justify the choice of native scripting (3.3.2) and finally the choice of React Native (3.3.3).

3.3.1 Native Applications vs. Cross-platform applications

Although native development achieved a better performance (regarding multiple platform development), the fact that each platform requires advanced knowledge for its development as well as the costs that this knowledge would require, multiple platform development was the best option. Moreover, it was important, for our purpose, to reuse code to attack more than a market. We can see in Table 3.1 a synthesis of the advantages and disadvantages of the two types of development.

[9] Titanium: http://www.appcelerator.com
[10] NativeScript: https://www.nativescript.org
[11] React Native: https://facebook.github.io/react-native/

Table 3.1: Advantages and disadvantages of native vs. multiple platform development

Type of development	Advantages	Disadvantages
Native development	• Best performance • Technological compatibility • Offline functionality • User experience	• Each platform requires advanced knowledge • Development and maintenance costs • Development time
Multiple platform Development	• Code reuse • Cross-platform code sharing • Possibility to use and access native APIs • Common languages, such as JavaScript • Lower cost	• Lower performance • Although it's possible to access and use the native APIs, there are limitations

3.3.2 Web Development vs. Hybrid Development vs. Cross-Platform Development vs. Native Scripting

In Table 3.2 a synthesis of the advantages and disadvantages of the various options of cross-platform development can be seen.

Table 3.2: Advantages and disadvantages of multiple platform development options

Multiple platform development	Advantages	Disadvantages
Web development	• Different platforms • Code Sharing • JS, CSS e HTML	• Requires Internet • Limitations on access to native components • Performance
Hybrid development	• Code Sharing • Native components	• Limitations on access to native components • *Webview*
Cross platform development	• Possibility of using programming languages like C, Java or C# • Native components	• Possibility of using programming languages like C, Java or C • Unable to reuse user interface • Can't reuse platform-specific features
Native Scripting	• Code Sharing • Native components • The code runs at run time	• Performance is affected by the interpreter

Looking to the future, code reuse and cross-platform code sharing was a must. In addition, access to native components specific to the platform in question was also a very important requirement as we believe that native or redirected applications for a particular platform have better results in the user experience. The web development of the application offered the possibility of being able to access the application of any platform, as long as there was a browser and an Internet connection. Hybrid development was more responsive to our needs, however, the fact that the application was embedded in a webview and the limitations to native features made us focus our attention on cross-platform development and native scripting. Although cross compiler offers very similar advantages to native scripting, for example full access to native platform components is possible, it is not possible to reuse the user interface or platform-specific features. Therefore, our choice refers to the type of development that will give us greater freedom of access to the native components while we will be able to reuse code.

3.3.3 Appcelerator Titanium vs. Native Script vs. React Native

Within native scripting we studied Appcelerator Titanium, Native Script and React Native, opting for the Facebook framework.

The Appcelerator Titanium framework, according to the official website itself and *Heitkotter et al.* [39], although it provides a completely free version, this same version is limited. These additional features are only available with subscription to the framework. In addition, *Heitkotter et al.* [39] raises pertinent questions regarding the framework community, making references to some topics with no answers for weeks. Titanium also requires some specific knowledge of the structure of the framework itself for better development. From the official site of the Native Script we can find execution times (milliseconds) in two specific situations, compared to a native approach. The first situation is the startup time (defined as the time it takes after opening the application until the first screen appears) of a blank application with a single button and the variable packaging time (defined as the transformation of the representation of a variable storage). After analyzing these same execution times, it is concluded that the Native Script, despite having very close execution times of the native approach in the packaging, at the start of the application the execution time is a little distant, as is shown in Table 3.3 and 3.4 and at Figure 3.1 and 3.2 from official website of framework [12]. The other framework within the native scripting category was React Native. Besides being based on the architecture of React.js and being open source, according to *Beshir* [43] and *Hansson et al.*[38] it behaves very well compared to native development, getting good execution times.

Table 3.3: Initialization time (milliseconds), adapted from official website of framework.

	Run 1	**Run 2**	**Run 3**
Native	768ms	774ms	759ms
NativeScript	1135ms	1129ms	1138ms

[12]Native Script: https://www.nativescript.org/blog/nativescript-and-xamarin

Table 3.4: Package time (milliseconds), adapted from official website of framework.

	Run 1	**Run 2**	**Run 3**
Native	111ms	105ms	108ms
NativeScript	674ms	672ms	670ms

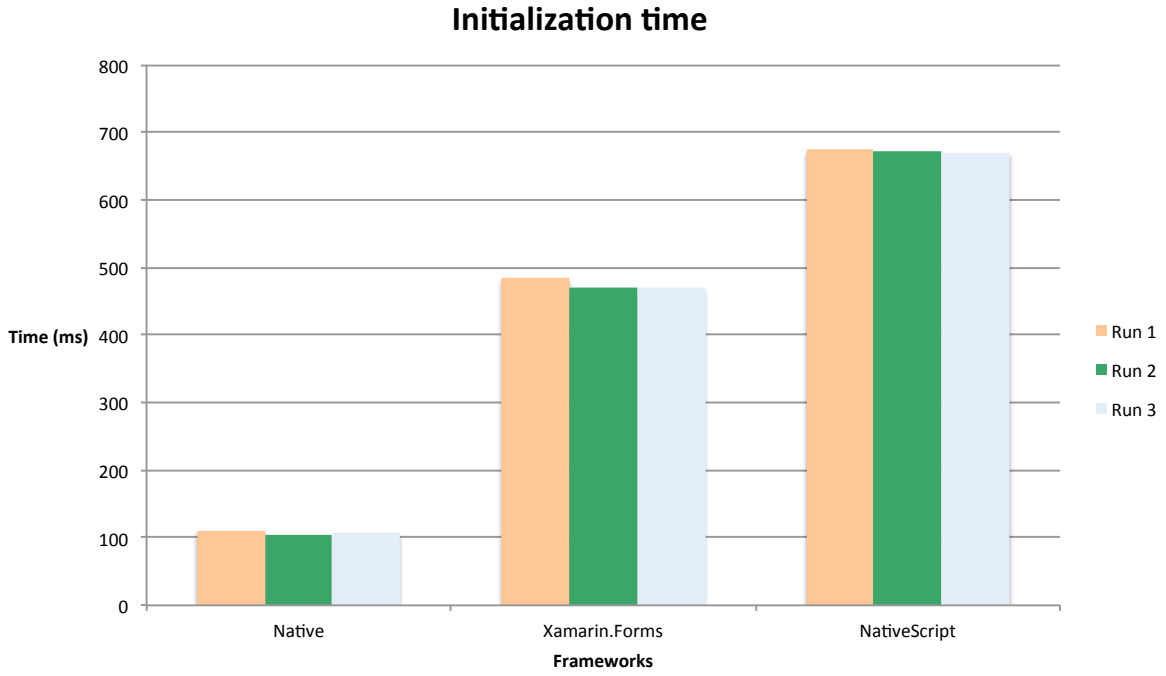

Figure 3.1: Graphic initialization time (milliseconds), adapted from official website of framework.

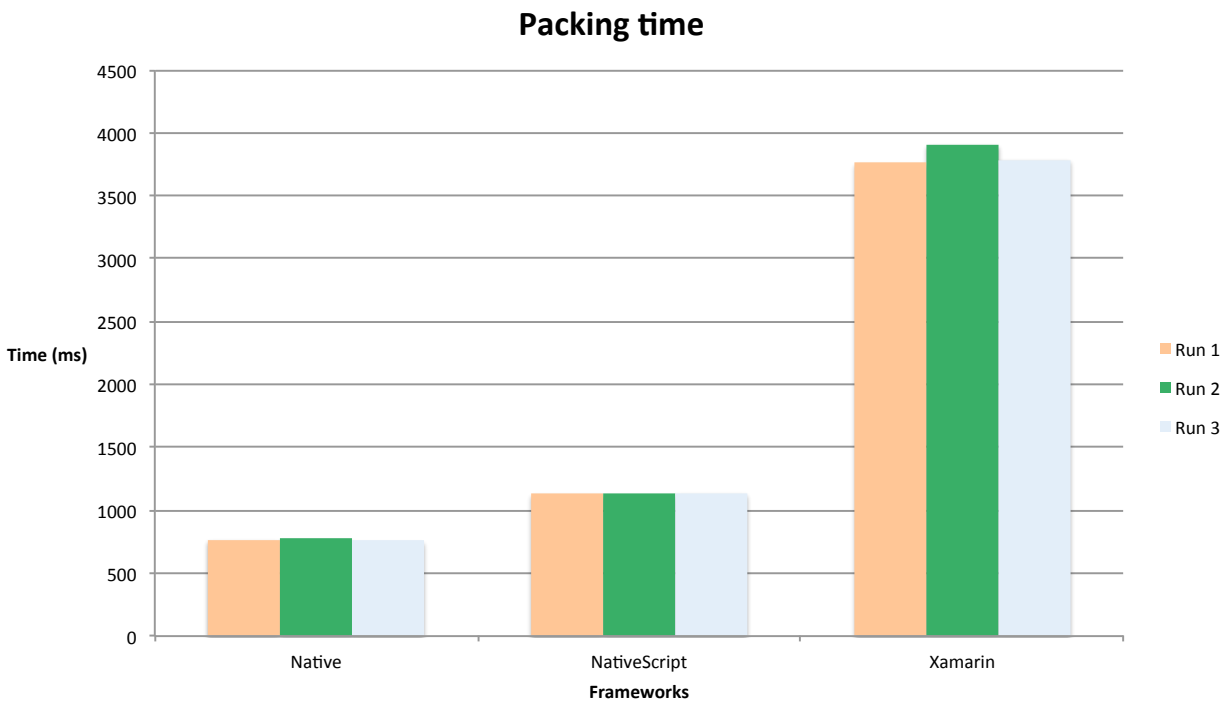

Figure 3.2: Graphic package time (milliseconds), adapted from official website of framework.

Of the three initial options, we first discarded Titanium, since in order to achieve a better development experience, we needed to get the subscribed version of the framework and to study the framework further. In addition, according to *Heitkotter et al.* [39] if the choice between Adobe Phone Gap and Titanium, and the native aspect of the application was not a factor to consider, the decision should fall on the Adobe Phone Gap.

Table 3.5: Synthesis of positive and negative aspects of Native Scripting.

Framework	**Important positive aspects**	**Important negative aspects**
Titanium	• Natives components and APIs	• Limitations on the free version • Advanced framework knowledge requirement • Community
Native Script	• Good packaging times compared to native development • *Open source* • Natives components and APIs	• Distant initialization times compared to native development
React Native	• React JS • *Open Source* • Virtual DOM • Natives components and APIs • Used on great sites like Facebook and Instagram	• Performance may be affected by code being run at runtime • Performance can be affected by the interpreter

When we analyze Table 3.5 of the synthesis of the three native scripting frameworks, we conclude that our choice would be between Native Script and React Native. The choice was the framework created and maintained by Facebook for the strong characteristics that differentiated it: architecture very similar to React.js, Virtual Document Object Model (VDOM) and rendering of isolated components. In addition, it is a recent framework with a good community and many

great sites already use it.

Chapter 4

React Native

React Native arises from the need for Facebook to give programmers a native experience while enjoying a React-like development experience. That is, React Native allows the native development of iOS or Android using a React-based architecture. Native development can take more time, however there are several reasons why we believe that it is possible to offer better experiences to users with native applications. If on the one hand we have access to platform-specific user interface components, on the other hand we have sophisticated gesture recognizers. In a first phase React Native only supported the development of iOS applications, being then expanded to Android development [38]. According to Facebook and *Hansson et al.* [38] both React Native and React are very similar in terms of code structure, however React operates on the Document Object Model (DOM) in a browser, while React Native Operates in the mobile application.

4.1 React.js

React is a JavaScript library, open source, widely used on major sites such as Facebook or Netflix. Since 2013, when Facebook officially introduced React, this platform has been growing sharply [39]. React gives programmers complete freedom and concentration in product development [39]. In addition, component by component is built, with each component representing a display. In this way, the components make it easier to change only a part of the application, thus creating a faster and easier development cycle, since constant recompilations are avoided. We can say that React is not a Model-view-controller (MVC)[1] framework, but rather a library that encourages the creation of reusable components. When developing in pure JavaScript, you need to make changes to the DOM to keep it up to date. However, React addresses the issue differently. Whenever a component is created or initialized for the first time, the rendering method is called, presenting a view. Then, because React creates a data-memory cache in memory, whenever a change exists, the rendering method calculates the resulting differences by displaying them. React does not run in the *textitbrowser* nor rendering is done through divs or spans, but at runtime, in JavaScriptCore, at the highest level. An important feature that distinguishes React

[1]Framework MVC: separates the information representation from the user interaction.

Native from React.js is how React Native behaves to update the changes made. React.js changes directly in the browser DOM, replacing the old content, then invoking a full re-rendering of the content. React Native recalculates the changes made and through Virtual Document Object Model (VDOM) it only renders the changes. In addition, VDOM is never rendered in the browser, saving time [43].

4.1.1 Component lifecycle

The lifecycle of the components can be debited in three groups [2]: mounting, unmounting and possible updates of the components [44][45]. Thanks to the component lifecycle, it is possible for the programmer to customize code at each rendering step, as we will describe next [44][45].

Mounting

constructor(object props): The constructor of a component is always called before it is mounted. However, if you do not initialize states or do not bind methods in the constructor (bind), you do not have to implement it.

componentWillMount(): This method is invoked only once, just before the mount of the component occurs, that is, before the render() call. The React Application Programming Interface (API) itself advises using the constructor() instead of the componentWillMount().

render(): The render() method is an indispensable method. Whenever it is called it examines this.props and this.state by returning a React component. However, it is possible for the render() method to return null or false, and in this case, nothing will be rendered. It is advised that the render() method does not modify the state of the component, because each time this method is invoked, it should return the same result. It is at this point in the life cycle of a component that the states and the values are interpreted, returning a certain output.

componentDidMount(): Immediately upon a component being mounted, componentDidMount() is invoked. This method is usually used either to load data (for example, to load remote data) or to initialize the DOM. This method is only invoked once after the first render. It is advisable to use this method for asynchronous calls or to run delayed code (setTimeOut()).

Update

componentWillReceiveProps(object nextProps): When a component receives new objects, the componentWillReceiveProps() is invoked. However, during mount of a component (initial rendering) this method is never called, because React invokes it only if some of its components can be updated, that is, they have received new objects. This method can be used, for example, to update the response depending on the comparison result of this.props with the nextProps. However, we must make sure that we are comparing the correct values, since React can call componentWillReceiveProps() at a time when the values have not yet been changed.

[2]The Component Lifecycle: https://facebook.github.io/react/docs/react-component.html

You can optionally use this.setState(), within this method, to update the internal state of the component before render() is invoked.

shouldComponentUpdate (object nextProps, object nextState): This method is invoked when new objects or states are received, before rendering. ShouldComponentUpdate () is never called in the initial render() or when forceUpdate() is used. The default value of this method is True, and whenever it returns as False, methods such as componentWillUpdate(), render (), or componentDidUpdate() are not invoked.

componentWillUpdate (object nextProps, object nextState): ComponentWillUpdate() is called immediately before rendering, when new objects are received(props or states). In other words, this method is invoked when the shouldComponentUpdate() returns True. When you need to update the state in this method, use the componentWillReceiveProps() instead of this.setState().

render (): When render() is invoked, it is assumed that the shouldComponentUpdate() returned True. The output of the render() must be a React or null element. In this later case it is used when you do not want to render anything.

componentDidUpdate(object prevProps, object prevState): When the component is updated you can use componentDidUpdate() to operate on the DOM.

Unmounting

componentWillUnmount (): Before a component is unmounted and destroyed, the componentWillUnmount() is invoked. This method is conducive to performing any necessary cleanup, canceling requests or simply cleaning up previously created DOM elements.

4.1.2 API Component

We will present important features and APIs of React.js too.

forceUpdate() By default, whenever the state or values of a component change, it is rendered again. However, if render() depends on other data, you can force rendering using forceUpdate(). When using forceUpdate(), render() is immediately called, without going through the shouldComponentUpdate(). Although forceUpdate() in some cases works, you should avoid using React's own API.

this.props We can pass certain properties to the vectors (props), being possible to access them either through the methods of the component or through this.props. This way, you can change the way the component is processed or the way the component behaves. However, it is a good practice to never change the props within the component's own methods.

this.state The state (user-defined and must be of obj type) contains specific data to a particular component that may change over time. For example, it is possible for the components themselves to maintain their inward states within the state of the object. You can access this.state

within the component's methods. You can never assign a specific key directly to a state, but you should use this.setState().

this.setState() The components themselves, as we have said, can update their state by passing an object to the this.setState() method.

4.2 Native

Recalling that a native application is an application specifically developed for a specific platform, with full access to native APIs, we will explain how React Native communicates with native APIs during application development. As can be seen in Figure 4.1 (adapted from [38]), React Native communicates with the native APIs through a JavaScript bridge, with JavaScriptCore as the interpreter. The function of the JavaScript bridge is to communicate and connect the JavaScript layer with the native side of the application, and the execution of the JavaScript layer is performed on a separate asynchronous thread, not interfering with the native user interface. When a call is made from the JavaScript layer side, that same call is stored in a message queue if the call can not be sent directly. The JavaScript bridge has the function of before sending information to the native layer side of converting all JavaScript data types to native data types. [38]

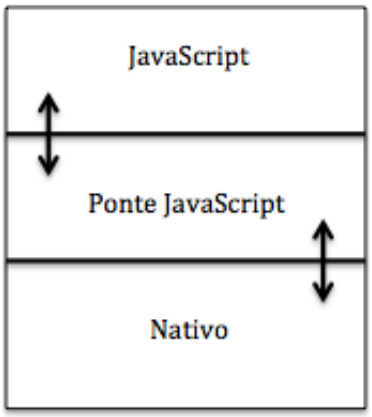

Figure 4.1: Example of communication in React Native.

4.3 Virtual DOM

The VDOM is updated whenever there is a component change. That is, whenever a change exists, VDOM compares the content with the browser DOM, calculating the minimum amount of changes needed to display the new changes. Then the changes are made, through a queue and a path, asynchronously, through the imperative manipulation. In React Native there is no need to compile the entire application whenever there is a change because a virtual hierarchy is used. [38]

4.4 JavaScript with a syntax extension (JSX)

When developing applications using React Native it is possible for the programmer to choose whether to use JavaScript or JSX, thus allowing a mixture between JavaScript and eXtensible Markup Language (XML) tags. However, React Native itself recommends using JSX, since code written in JSX is optimized to run faster than if it were written in JavaScript. In addition, React Native provides a compiler that supports JSX, called Babel9 [38] [43].

4.5 Libraries

In addition to React Native components (View, Text, etc.), there are many open source component libraries. These libraries vary in both size and purpose. Next we will present all the libraries that we use in application development.

React Native Camera [3]: This library has a specific method (*OnBarCodeRead*) that is called whenever a barcode is detected in the camera. The data is stored in an object that contains both the bar code value and the bar code limits.

React Redux[4]: Redux allows us to store objects in the application state (store), making all the modifications that we want to do are through pure functions (reducers receive actions), generating a new state (the state is immutable).

React Redux Thunk[5]: Through the Redux Thunk middleware it is possible to create actions that return functions, instead of a simple action. Thus, through this library, store values returned by an asynchronous function can be stored in the Store.

React Native FS[6]: It is from this library that we both read files and store files on the user's phone. More specifically we need to read the Assets [7] questionnaires (*readFileAssets*), save the files to serialize both the user and the responses locally (*appendFile*), and finally we need to read the files that are locally stored (*readFile*). To save files to the user's phone we need the absolute path to the external files that is provided by the *ExternalDirectoryPath* constant (string) provided by this library.

ReadFileAssets This method allows you to read the files that are in Assets by simply entering the name of the file. The encoding used by default is UT8, but you can use another one within the options: ascii or base64. At the end of the file reading the contents of the file or the generated error is returned. You can only use *readFileAssets* on Android.

ReadFile With *readFile* it is possible to read the files in a certain path, returning their

[3] React Native Camera:https://github.com/lwansbrough/react-native-camera
[4] https://github.com/reactjs/react-redux
[5] https://github.com/gaearon/redux-thunk
[6] https://github.com/itinance/react-native-fs
[7] ttps://developer.android.com/reference/android/content/res/AssetManager.html

contents.

AppendFile Adds the content sent as a method parameter to the previously specified path.

WriteFile Writes the submitted content to the specified path.

React Native i18n[8]: From the locale attribute of this library it is possible to know the location of the device, which is very useful either in terms of statistics (logs) or to offer the user a questionnaire in the English language if it is justified (only available in English and Portuguese).

React Native Radio Buttons[9]: This library implements the behavior of radio buttons. That is, they allow several options for answers, but only one option selected at a given time. It is possible to customize both the component and the options, which gives the programmer plenty of autonomy.

React Native Slider[10]: Similar to the behavior of the last library described, React Native Slider lets you customize the normal behavior of a slider.

React Native Progress[11]: With the help of this library you can implement a progress bar. Easily customize colors, sizes, animations and bar boost function.

React Native Background job (only Android)[12]:

From this API it is possible to run Javascript code even when application is closed or in the background.

React Native Background Timer[13]:

From this API it is possible to run Javascript code even when application is closed or in the background.

[8] https://github.com/AlexanderZaytsev/react-native-i18n
[9] https://github.com/ArnaudRinquin/react-native-radio-buttons
[10] https://github.com/jeanregisser/react-native-slider
[11] https://github.com/oblador/react-native-progress
[12] https://github.com/vikeri/react-native-background-job
[13] https://github.com/ocetnik/react-native-background-timer

Chapter 5

Application Development

This chapter contains all the details about application development. In the first subchapter (5.1), some basic ideas about the application will be recalled, as well as some important notions about the Android and iOS operating systems. Then, the rest of the chapter is dedicated to the development of the application, presenting its architecture with the respective workflows.

5.1 Introduction

React Native was the chosen framework for application development, as we explained in Chapter 3 and detailed some important features in Chapter 4.

We must not forget that our users are ill and undergoing complex treatments, so they are people with less physical or emotional availability. The data resulting from the users' responses to the questionnaires through our application are sensitive, so both security and robustness were characteristics that worried us from the beginning. In this chapter, we'll look at all the important aspects of implementing the application, emphasizing both the authentication part and the sending part of the responses from the questionnaire to the server. We emphasize these two implementations simply because it is in these parts that we receive or send confidential patient data (patient responses to the questionnaire, for example).

First let's talk about some important things about Android and iOS, how did [38].

5.1.1 Android

Android[1] is a Linux-based open-source operating system that is available for many device types. In other words, we can define Android as a software stack or a stack of programs grouped in layers, as we can see in Figure 5.1[2].

[1] https://developer.android.com/guide/index.html?hl=pt-br
[2] https://developer.android.com/guide/index.html?hl=pt-br

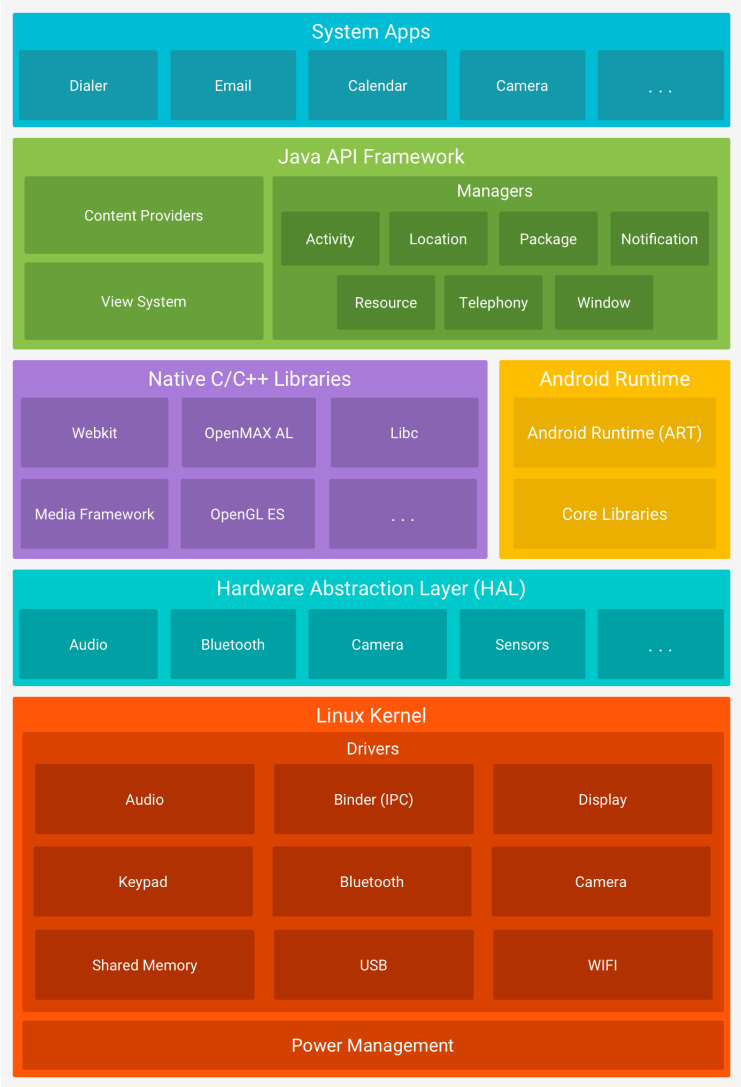

Figure 5.1: Android architecture from Android website official

In the next subsection we will explain each of the Android stack layers[3].

Android Architecture

Let's explain the Android architecture.

- **Linux Kernel:** At the bottom of the stack is the Linux kernel, which is responsible, for example, of the memory management programs or security settings.
- **Hardware Abstraction Layer:** It is through this layer of hardware abstraction that it is possible to access specific hardware components of the Android system. For example, when it is necessary to access device hardware, Android loads the library module for this layer.

[3]https://developer.android.com/guide/platform/index.html?hl=pt-br

- **Android Runtime:** For Android versions equal to or higher than version 5.0, each application runs its own process with an own instance of Android Runtime (ART). Android Runtime is the runtime manager used by both applications and Android system services. The fact that each application executes its own process makes the applications independent of each other, that is, if an application fails nothing will affect the remaining applications.
- **Native C / C ++ libraries:** On Android there are several components implemented by native code. It is in the library layer that one can find a set of instructions that tell the device how to deal with different types of data.
- **Java API Structure:** All features of the Android operating system are available on this stack layer. Programmers have full access to this layer because these features are essential for building applications for this platform.
- **System Applications:** It is in this layer that there is interaction between the user and the mobile device in a wide range of applications in the system: e-mail, SMS sending, calendar, browser, contacts, among many others.

Application structure

Android applications are created using different application components. These application components can be divided into activities, services, content providers and transmission receivers.[4] Each of these components has a specific function and life cycle.

- **Activities:** Activities represent a single screen with a user interface. That is, each activity should function independently and be able to be started from another application. For example, after taking a picture using the device's camera, the camera application must allow you to start activity in the email application and create a new email by attaching the photo. An activity has three different states: "resume", pause and stop. A "resume" activity is activity that is running in the foreground on the device, so it is current activity. Already a paused activity is an activity that is running in the background. Finally, a stopped or interrupted activity is an activity that is not visible on the screen but exists in memory.
- **Services:** This component does not represent an interface for the user, since they are executed in the background (for example downloading a .pdf while reading the news in the browser). As with Activities, Services are started by other components.
- **Content providers:** Content providers are responsible for storing data. Other applications, through content providers, can access and even modify this data, if the content provider allows it. An example of a content provider is the contact management system on the Android system.
- **Transmission Receivers:** This component is responsible for transmitting messages when, for example, the mobile device is low battery. Although broadcast receivers do not interact directly with the user interface, they can create notifications in the status bar.

[4] https://developer.android.com/guide/components/fundamentals.html?hl=pt-br

5.1.2 iOS

IOS is an operating system structured in four different layers one on top of the other: Cocoa Touch, Media, Core Services and Core OS.[5] Figure 5.2 represents the layers of the iOS framework.

In the next subsection we will explain each of the layers of the iOS operating system.

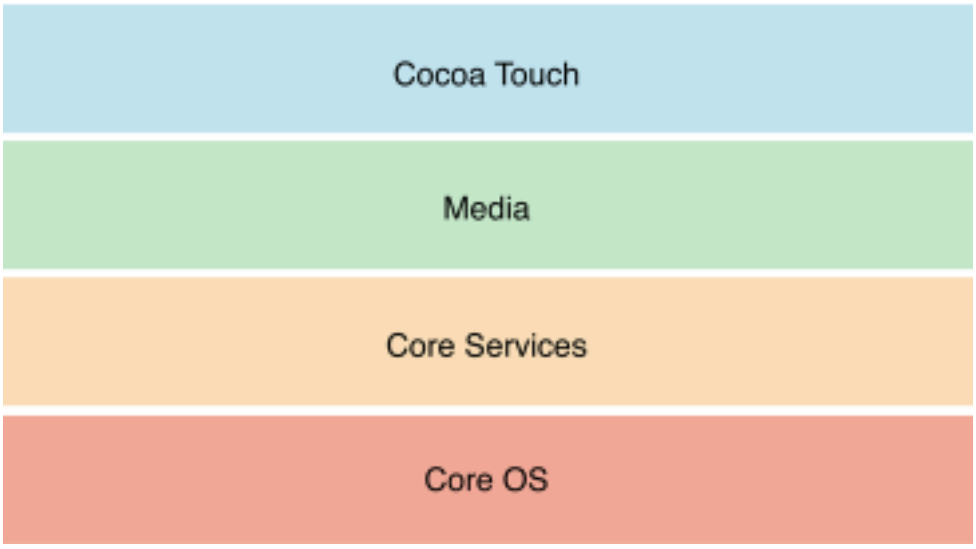

Figure 5.2: iOS architecture from iOS website official.

iOS Architecture

Let's explain the iOS architecture.

- **Cocoa Touch:** This layer contains frameworks for creating iOS applications. These frameworks provide the application look and feel, the basic infrastructure, and the high-level system services. Apple recommends that developers only use frameworks on this layer.
- **Media:** When the programmer needs multimedia content, sound, graphics or video should focus on this layer. This layer provides frameworks that encapsulate complex tasks, providing more accessible APIs.
- **Core Service:** It is in this layer that are the fundamental services (network, location, etc.) for the applications.
- **Core OS:** Finally, Core OS is the low-level features that other structures encapsulate and use.

Application structure

Apple recommends that an iOS application respect the Model-view-controller (MVC) architecture[6], since this structure separates the data, the view, and the controller. In this way, this

[5] https://developer.apple.com/develop/
[6] https://pt.wikipedia.org/wiki/MVC

architecture facilitates for example the treatment of different resolutions and screen sizes of the devices, since only it is necessary to change in the view component, without affecting the other components. You can see the representation of the MVC architecture in Figure 5.3.

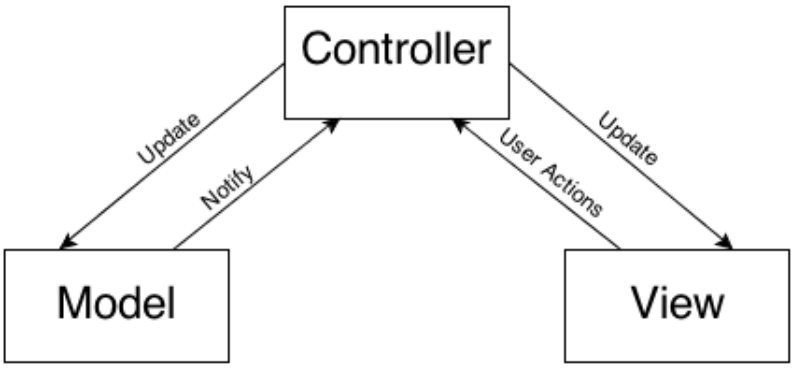

Figure 5.3: Model View Controller architecture.

The model manipulates the data and notifies the controller when data changes. The view is the visual representation of the contents and / or data in an application. Therefore, the view is who controls the presentation of data, notifying the controller whenever there is action on the part of the user. Finally, the controller acts as a mediator between the model and the view.

Application state

An iOS application can have the following statuses: not running, inactive, active, background, suspended. Figure 5.4 shows the possible transitions between states in iOS.[7]

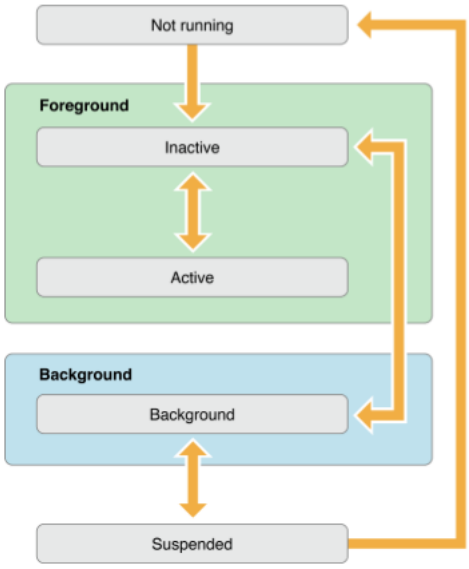

Figure 5.4: Possible transitions between states in iOS from iOS website official.

- **Not running:** Whenever an application is not running, it will not execute any code as it

[7]https://developer.apple.com/develop/

has not been started or closed by the system itself.

- **Inactive:** The application is running in the foreground, however it is not receiving events right now. Usually, an application stays in this state very little time.
- **Active:** The application is running in the foreground and is receiving events.
- **Background:** The application is in the background running code. Often, when an application enters this state, it is to be suspended. When an application runs directly to the background, it enters directly into this state and not into the inactive state.
- **Suspended** When the application is in the background without running code, the system automatically moves application to this type of state, not notifying the user. When an application is suspended even though it does not execute code it is in memory.

5.2 User Stories

In this sub chapter we will show you how we have idealized our solution. Once we have presented all the requirements of our solution, we will present both the detailed characterization of the various types of actors and the key functionality of the solution.

5.2.1 Roles

Our solution will have 4 types of actors.

- **User:** The actor "User" is the patient himself.
- **Manager:** The actor "Manager" is the nurse.
- **Super Manager:** The actor "Super Manager" is a type of actor that can manage either the "Manager" or the "User". This type of actor can be associated with a nurse who manages all the others.
- **Admin:** The actor "Admin" manages all the users present in the solution.

5.2.2 Use cases

In order to better understand the interaction of the actors and the functionalities that each one can perform in our solution, we create four diagrams corresponding to each one of the actors defined in 5.2.1.

The actor "User" can answer the questionnaire and validate the answers. In this case, the only interaction initiated by the application is the notification to the "User" informing that there are questionnaires for answering. It's possible to see this on Figure 5.5.

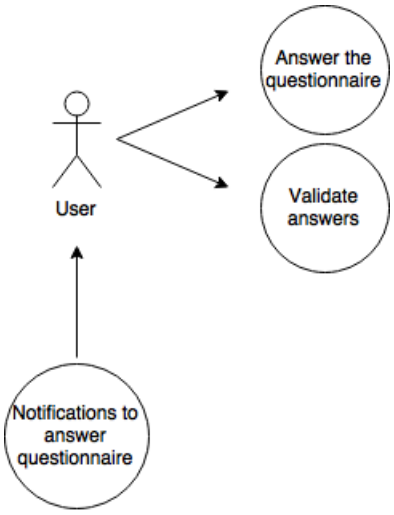

Figure 5.5: "User" interaction.

The actor "Manager" can authentication actors of type "User", answer the questionnaires as the "User", validate answers, define questionnaires for the "Users", access to data of their users and authentication of self smartphone. The application interacts with this type of users with notifications of the users associated with a specific manager. However, "Manager" can disable notifications. It's possible to see this on Figure 5.6.

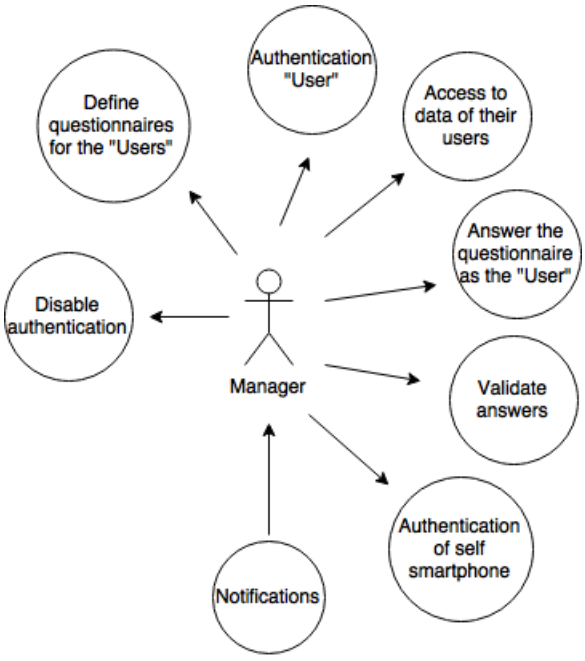

Figure 5.6: "Manager" interaction.

The actor "Super Manager" can do all the actions described in the actor "Manager" plus create actors of type "Manager".It's possible to see this on Figure 5.7.

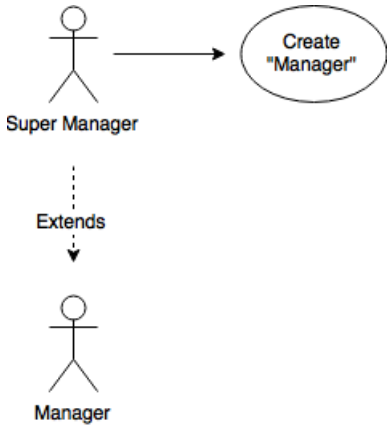

Figure 5.7: "Super Manager" interaction.

The actor "Admin" type can do all the actions described in the actor "Super Manager" plus create actors of type "Super Manager" and access to data. It's possible to see this on Figure 5.8.

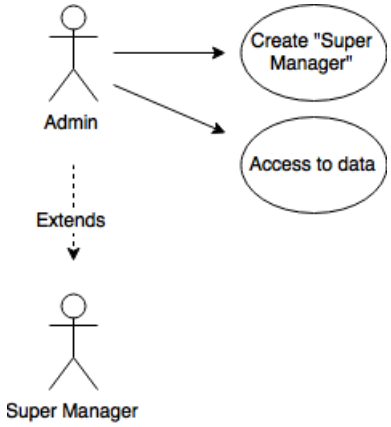

Figure 5.8: "Admin" interaction.

5.2.3 Scenarios

Then, to better understand the operation of the solution, we created two key scenarios that illustrate all the interaction between actors and the application.

New user of the application (This example is presented in Figure 5.9.)

- **1** One person, Gonçalo, goes to Google Play[8] or the App Store[9], depending on the operating system of your mobile device, and downloads the application.
- **2** The application is installed on your smartphone.
- **3** Gonçalo opens the application.

[8] https://play.google.com/store?hl=pt-PT
[9] https://itunes.apple.com/pt/genre/ios/id36?mt=8

- **4** Application check if token exists or not (as it is first register there is no token).
- **5** Gonçalo requires an authentication to the Manager.
- **6** Manager requires a QRCODE to the server.
- **7** Server provides a QRCODE to Gonçalo.
- **8** Capture of the QRCODE.
- **9** Application sends QRCODE data to server.
- **10** Server sends a token to the application.
- **11** Application asks for user data.
- **12-** Server responds with data.
- **13** Application saves data (token plus user information).
- **14** Authentication done.
- **15** Gonçalo starts answering to the questionnaire.
- **16** Application notifies the user that it has reached the end of the questionnaire.
- **17** Gonçalo validates his answers.
- **18** Application sends the answers to the server.
- **19** Server informs the user that it has received the request to send the answers and as soon as possible the answers will be sent.

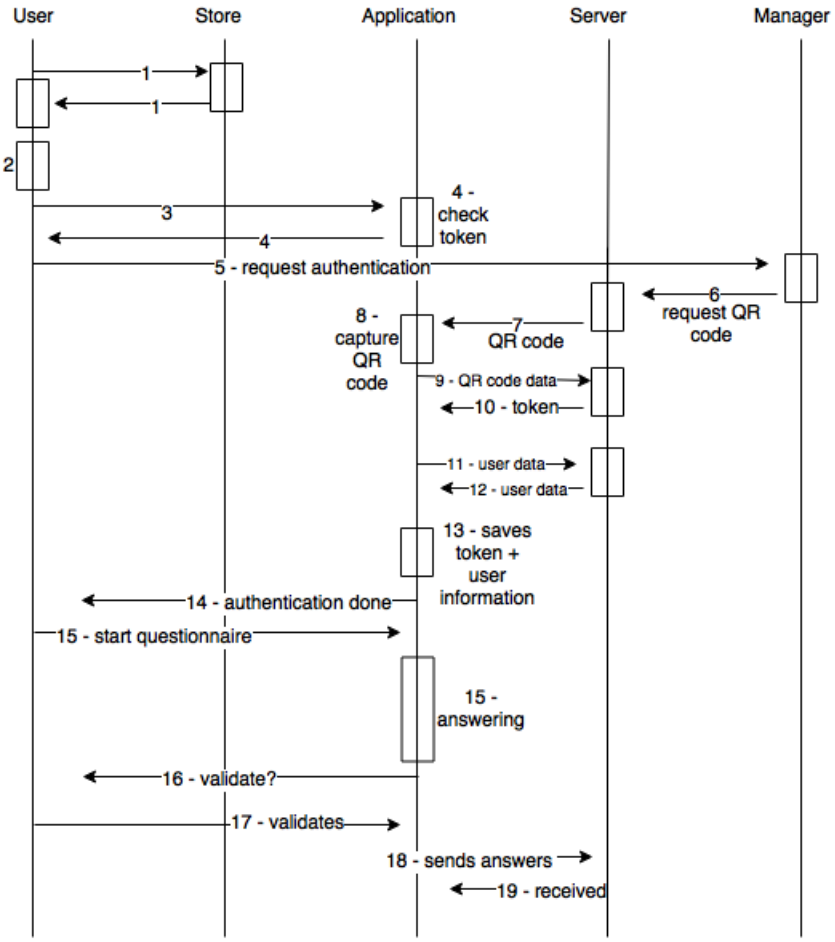

Figure 5.9: Unknown person of the application interaction example. (The numbers present in the figure actions are describe above.)

Person unable to respond to the questionnaire (This example is presented in Figure 5.10)

- **1** Manager opens the application.
- **2** Application asks the user data to manager
- **3** Manager ask for the data (ID and pathology) to the user.
- **4** User provides the data.
- **5** Manager inserts the data into the application.
- **6** Application notifies the Manager that he can start the questionnaire.
- **7** Manager starts the questionnaire.
- **8** Manager completes the questionnaire as the user: reads the questions and asks the user for answers.

- **9** Application notifies the manager that it has reached the end of the questionnaire.
- **10** Manager notifies the user that it has reached the end of the questionnaire.
- **11** User validates his answers.
- **12** Manager validates user answers in application.
- **13** Application sends the answers to the server.
- **14** Server informs the manager that it has received the request to send the answers and as soon as possible the answers will be sent.

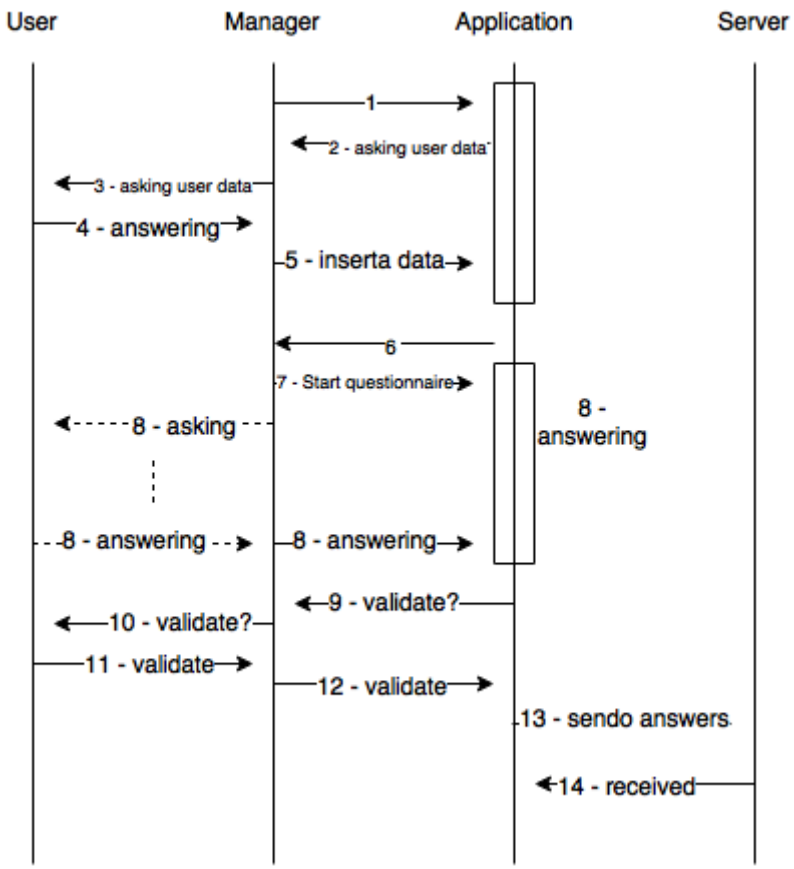

Figure 5.10: Person unable to respond to the questionnaire interaction example. The numbers present in the figure actions are describe above)

5.3 Application Architecture

Let's begin by presenting both the overview and the overall workflow of our solution, and in the next subsection we will go into detail in state of the application.

5.3.1 Overview

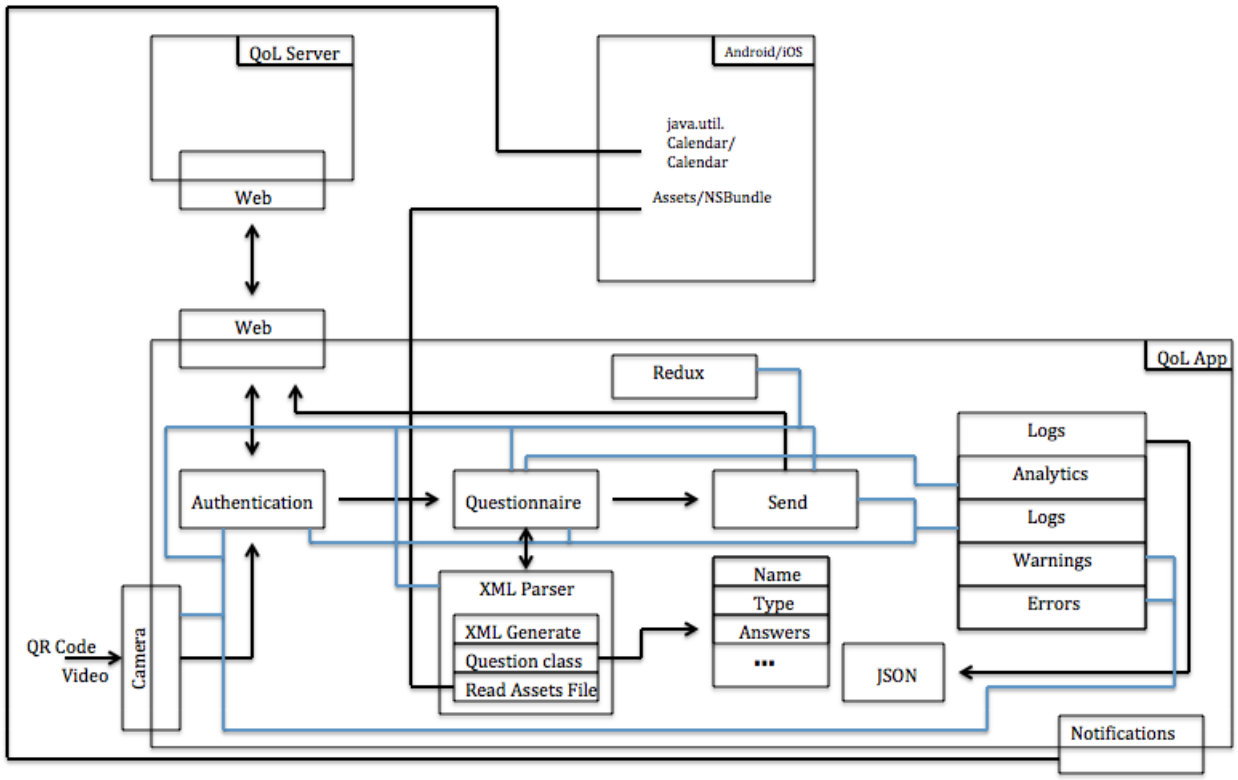

Figure 5.11: Architecture of Application.

As we can see in Figure 5.11 our solution is divided into 5 main components: Authentication, Questionnaire where we include the XML Parser, the Logs, the state of the application itself and, finally, sending the responses to the server.

5.3.2 General workflow

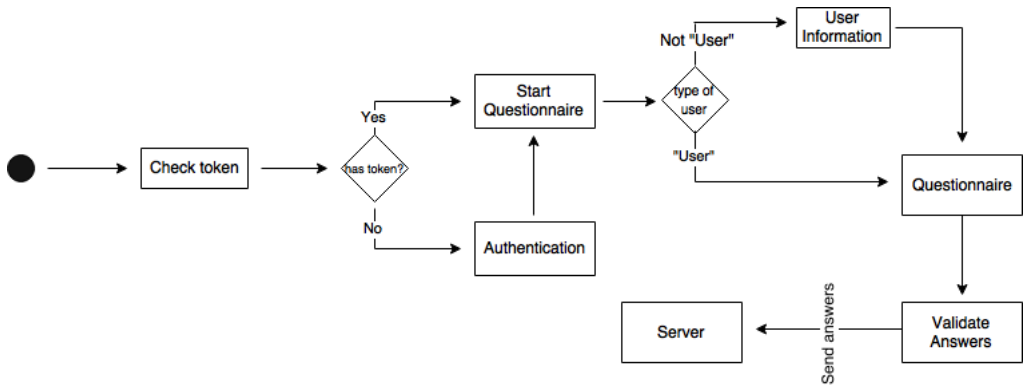

Figure 5.12: General application workflow.

When entering the application, the first thing that the application does is to see whether or not there is already a previous authentication. For this the application sees if there is a token (token is user authentication) already created previously with the data of the patient. Depending on the response, the user is forwarded or authenticated (if they have never authenticated) or is invited to start responding to the questionnaire. In the end, after having already answered the questionnaire, the user will only have to send the answers, that is, to complete the questionnaire. The application will send the responses to the server as soon as you have Internet available. At the same time that the application is noticing whether or not an authentication already exists, the user type is saved in the application state, if it already exists. This information is very important because there are two possible flows depending on the type of user. Users of type "User" is when patients themselves respond to the questionnaire. In this case, the user who is not logged in to the application is invited to start responding to the questionnaire. In contrast, users of the "Manager" type will have to fill in some information (ID PATIENT + PATHOLOGY) regarding the user that will authenticate and administer the questionnaire. The general application workflow is represented in Figure 5.12.

5.3.3 Application State

It is very important that we store certain data in the state of the application. For example, we need to pass information from one component to another, but we also need to communicate with the server.

React is a JavaScript library that helps us split an application into multiple components, but does not specifically specify how to track states and how to handle all events correctly. Either to save the state of the application or to modify it, we use Redux and Redux Thunk. [10] [11] [12] [13] [14] This last one to save data in the state coming from asynchronous calls.

Redux

Redux is an open-source library that handles states and events. The state of every application is stored in a tree of objects inside a store. The only way to change states is to issue an action (ie describing what happened). Reducers are used to specify how the state tree is transformed by actions.

State shape

The state of our application can be seen in Figure 5.13 and the explanation of each variable immediately following.

[10] https://medium.com/@stowball/a-dummys-guide-to-redux-and-thunk-in-react-d8904a7005d3
[11] https://medium.com/react-native-training/redux-4-ways-95a130da0cdc
[12] http://redux.js.org/docs/advanced/Middleware.html
[13] http://www.thegreatcodeadventure.com/react-redux-tutorial-part-iii-async-redux/
[14] http://blog.nojaf.com/2015/12/06/redux-thunk/

```
const initialState = {
   current : 1,
   answers : {},
   questionnaire : {},
   patient_id: null,
   startDate: null,
   endDate: null,
   role: null,
   quest_id: null,
   error: null
}
```

Figure 5.13: State of application.

- **current:** Current question. It starts by default at 1. It is only used to help navigate the questionnaire.
- **answers:** Object of type index: answer. Where answer is the answer to the corresponding index question.
- **questionnaire:** Save the questionnaire.
- **patient_id:** Save the patient id if you are not a user.
- **startDate:** It saves the date that the user began to respond to the questionnaire.
- **endDate:** Saves the date that the user has just completed the questionnaire.
- **role:** Type of user.
- **quest_id:** Questionnaire id.
- **error:** Information regarding possible errors.

Actions

Object that describes how we want to transform the state. It is the only source of information for the store (available through store.dispatch ()). Action creators are used to create action. Type must be a unique name that clearly indicates the intent of the action.

For example, our action creator to deal with the user's event answering a question in the questionnaire is:

```
export const setAnswer = (index, answer) => ({ type: 'SET_ANSWER', index,
    answer });
```

Thus, we are giving the information to reducer that we have a new answer "answer", corresponding to the question number "index" of the questionnaire.

Reducers

Function that receives a previous state, a certain action and returns the next state. The main goal is to understand the actions already defined and to execute them. When there are no changes, it returns the same state. In no case should states change directly, as it would imply changing the history of the state.

The Reducer that receives the action 'SET_ANSWER' and creates a new state is:

```
export const reducer = (state=initialState, action) => {
  switch (action.type) {
case 'SET_ANSWER':
    return Object.assign({}, state, { answers: { ...state.answers,
        [action.index]: action.answer } });
default:
    return state;
  }
}
```

Store

Contains the status of the application. Allows state access via getStore (). It also allows updates from dispatch (action).

The main reason for Redux not dealing with asynchronous calls is the very definition of reducer. Recall that a reducer is a function that receives the current state of the application and an action, returning the next state. In other words, we can say that it is a pure function because it does not modify the current state. When we made an asynchronous call with Redux we would have to wait until a Promise was resolved and then return to the next state. We could even create an action for the beginning of Promise and one for when it was resolved, the problem was when Promise was rejected. Also, addressing the problem in this way would be to decline the definition of reducer. This is how Redux Thunk comes in.

Redux Thunk

Redux Thunk allows actions creators instead of returning only one action to return a function. You can use Thunk to delay sending an action or test a certain condition before making a dispatch.

Middleware

For Redux to support asynchronous calls it is necessary to use Middleware. Middleware is nothing more than a piece of code that is placed between the structure that receives an order and the structure that generates a response.

To use the advantages of Redux Thunk we just have to add an argument to the creation of the store:

```
Let store = createStore (reduce, applyMiddleware (thunk));
```

5.4 Authentication

In this sub chapter we will present the authentication workflow, the protocols used and some examples.

Workflow

In case the user has to authenticate, that is, there is no token, just use the camera of your device and take a photo to the QR Code gived by Manager for example. After the application exchanges the QR Code with a registration token associated with the user with various information relevant to the application (for example the type of user), it will be forwarded to the component that starts the questionnaire. In contrast, an already authenticated user will be invited to start responding to the questionnaire. The authentication workflow can see on Figure 5.14.

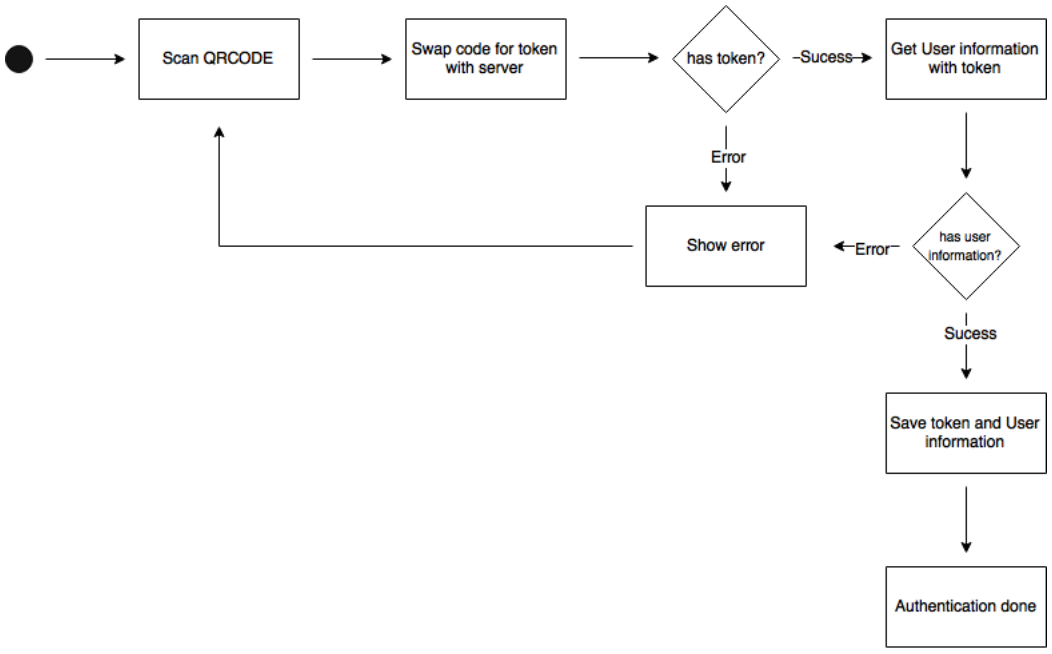

Figure 5.14: Authentication workflow

Used protocols

As we have already mentioned, the security and robustness of data exchange between application and server is very important. In order to implement the authentication of the application as described above and as shown in Figure 5.14, we studied RFC6749[15] and RFC 6750[16].

[15] https://tools.ietf.org/html/rfc6749
[16] https://tools.ietf.org/html/rfc6750

RFC6749

Auth 2.0 allows a third-party application to have limited access to an HTTP service. We use this protocol when we ask the server for a valid token, sending a QR Code. In the order we have some interesting attributes:

- **grant_type:** Required and value must be set to "permission_code".
- **code:** It is mandatory and in our case is the data in the QR Code that the user just read through the camera of your mobile device.
- **client_id:** Required whenever the client is not authenticated with the server.

After the request is made, the server must require client-to-client authentication; ensure that the authentication code has been issued; verify the validity of the authentication code and ensure the "redirect_uri" parameter. If all goes well the client is authenticated (200 Ok) and receives an access token with the following parameters:

- **acess_token:** The access token issued by the authorization server.
- **token_type:** The token type issued by the authorization server. This value is case sensitive.

RFC 6750

This RFC describes how to use tokens in a way that protects them from misuse. The Authorization parameter is the token type set plus the token itself, and the request can only be made after confirming that the token type is "Bearer" on the client side.

We validate the token type as follows:

```
if (!/[\w\-.~+\/]+=*/.test(token)) {
      throw "invalid_bearer_token";
  }
```

Example

The authentication workflow is present in Figure 5.14 From this we will give a series of communication examples between the application and the server, for each of the authentication steps.

Scan QR Code

Through the React-Native-Camera API, the application offers its users a fairly easy way to capture a QR Code during the authentication process. When the QR Code is captured by the user it is saved to be used in the next authentication step.

Swap QR Code for token with server

After a QR Code already exists, the application places a request to the server to exchange the QR Code sent by a valid Token to the user who is doing the process. An example of an exchange request:

```
    POST /token HTTP/1.1
    Host: HOST
    Content-Type: application/x-www-form-urlencoded
    Accept: application/json
    User-Agent: APP_NAME/APP_VERSION

grant_type=authorization_code&client_id=1&code=QR_CODE_DATA
```

The response from the server will have to be something similar to that:

```
    HTTP/1.1 200 OK
    Date: Mon, 24 Apr 2017 10:15:16 GMT
    X-Powered-By: qolws/1.0.0-SNAPSHOT
    Pragma: no-cache
    Cache-Control: no-cache, no-store, must-revalidate
    Content-Type: application/json;charset=UTF-8
    Content-Length: Content-Length

    {
        "token_type":"bearer",
        "access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJxb"
    }
```

Then the token type returned by the server has to be validated on the application side. Thus, that the validated token application can obtain information relative to the user through that same token.

Get user_info with token

You must make another request to the server to get all the user information available. An example of requesting information from the user to the server:

```
   GET /api/user_info HTTP/1.1
    Host: HOST
    Accept: application/json
    User-Agent: APP_NAME/APP_VERSION
    Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJxb
    {
        quest_id: "qol5_v1",
        quest_dates: ["2017-04-24", "2017-05-24", "2017-07-23", "2018-01-25"],
        role: "user",
        creation_date: "2017-04-24"
    }
```

Save token and user_info

Lastly, both the token and all the information returned by the server are saved to the user. This information, for example, is useful for the application to know which questionnaire the user will have to respond to.

5.5 Questionnaire

Our application is ready to read any type of questionnaire that contains some of the widgets we defined (radio button, slider, checkbox or toggle) and respect the structure defined in XML format. Questionnaires are defined in XML to make it easier and more accessible for doctors to edit and read it.

The questionnaires are stored in the application files. When a user is authenticated the application knows which questionnaire this will answer and reads the questionnaire through this information (parser).

Structure of the XML questionnaire

The questionnaires are read by the application in XML format and have the following basic structure:

```
<questionnaire id="id">
   ...
   <question id="id" ask="ask" type="typeofask">
      <answer id="id">answer</answer>
                  ...
      <answer id="id">answer</answer>
   </question>
   ...
</questionnaire>
```

This structure can be affected depending on the widgets required by the questionnaire, as we will see some examples in the next subsection.

Either the ids of the questionnaires or the questions have to be authentic. This means that there can not be more than one questionnaire with the same id and, in the same questionnaire, there can be no question with the same ids.

We defined this structure so that we could use our solution with other questionnaires and even in other areas.

Widgets

As we have already mentioned, questions can have four types, and depending on the same type, there are slight variations in the attributes of the <question> tag. However, whenever necessary,

it's can add more widgets to the application (respecting the structure of the questionnaires defined by us).

During the tests as we perceive acceptance of the patients, it's possible to change the colors, format, among other things in the widgets. So it's possible at the same time that we test our application to improve it.

Radio button

In this type of widget the user can only select a answer, so it is used when we want to limit the user's answer. In the XML questionnaire we will have to put "Radiobutton" in the "type" of the "question" tag and use the "answer" tag to wrap each question answer option:

```
<question id="id" ask="ask" type="radiobutton">
      <answer id="id">answer</answer>
                ...
      <answer id="id">answer</answer>
   </question>
```

Checkbox

Unlike "Radiobutton", this type of widget allows the user to select more than one answer. Therefore, we can say that this widget is antithesis of the previous one:

```
<question id="id" ask="ask" type="checkbox">
      <answer id="id">answer</answer>
                ...
      <answer id="id">answer</answer>
   </question>
```

Slider

Sometimes it is necessary to know how the user feels at a given moment or how he or she would evaluate their health in the last week. It is at these times that the slider is used in the questionnaires because, in this way the user is obliged to find the best level, within limits, to describe his answer. The structure of question is:

```
<question id="id" ask="ask" type="slider" min="min" max="max"
    scale="scale"/>
```

In "Slider" questions it is necessary to define both the minimum value and the maximum value of the slider, as well as the desired scale, as we can see in the example above.

Toggle

Instead of asking the user to evaluate their health over the last week through a "Slide" level, it is possible to ask something like "Does this week feel better than the previous week?" In this

case, the answer we want in this attribute can be either “yes” or “no”, depending on the type of the attribute, and the value of the attribute is 0 or 1. However, we can also put our own answer options in this type of questions (using the catch "flag"):

```
<question id="id" ask="ask" type="toggle" flag="flag"/>
```

Reading the questionnaire

The reading of the questionnaire is done when we identify the type of user that we have and which questionnaire he/she will respond. After we use the React Native FS *readFileAssets()* method to read the Android Assets eXtensible Markup Language (XML) questionnaire file, we use the XML Document Object Model (DOM) and a class Question, previously defined by us, to successfully parse the questionnaire.

XML DOM

Assuming that our XML questionnaire is well-formed, we use the DOM library to read the entire questionnaire (XML file) and represent it in a node tree (xmlDoc)[17], as you can see a small example in Figure 5.15 of a questionnaire with only one question.

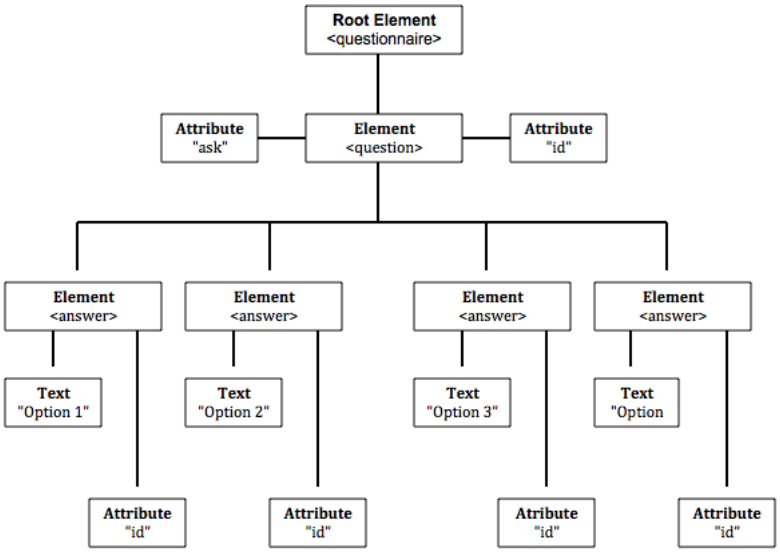

Figure 5.15: Example of node tree.

DOM

The DOM defines a standard for accessing and manipulating documents. [18] The DOM provides some very interesting properties:

[17] https://www.w3schools.com/xml/dom_intro.asp
[18] https://www.w3schools.com/xml/dom_intro.asp

- **object. nodeName :** Name of object.
- **object. nodeValue :** Value of object.
- **Name of object :** The parent node of object.
- **object . childNode :** The child node of object.
- **object . attributes:** The object nodes of attributes.

The DOM also provides some methods:

- **object . getElementByTagName(name) :** Get all elements with the tag "name".
- **object . appendChild(node) :** Inserts a child node in the object.
- **object . removeChild(node) :** Removes a child node from the object.

Question Class

As we process xmlDoc we are filling in a list of questions, where each position has the attributes defined, as we can see in the Figure 5.16. This class will be very useful for later putting the widget in the user interface, because all information regarding each question is stored in the class.

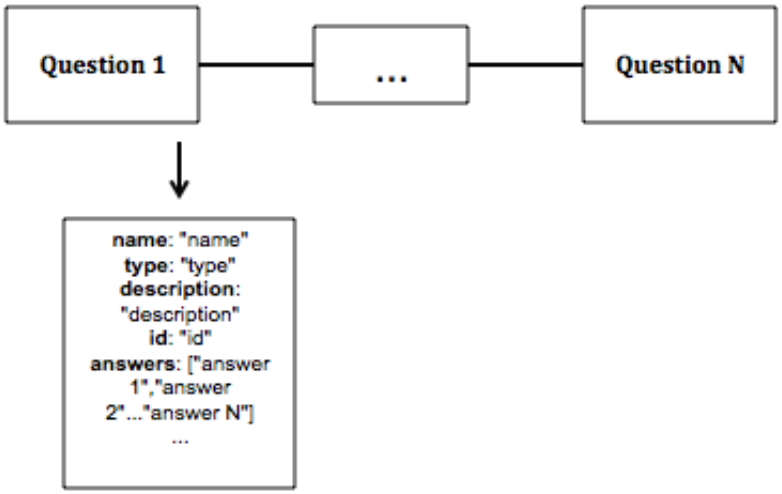

Figure 5.16: List of questions.

- **name :** Question.
- **type :** Type of Widget.
- **description :** Description of the question.

- **id :** Unique identifier of the question.
- **answers[] :** Options for answering the question. Each option is saved in a position of answers.
- **id_answers[] :** Each response identifier is saved in an id_answers position.
- **min :** Slider minimum value. It is only used in the Slider type widget.
- **max :** Slider maximum value. It is only used in the Slider type widget.
- **scale :** Slider scale value. It is only used in the Slider type widget.

Parser

After we have the xmlDoc filled out with the DOM tree of the questionnaire, our parse will process the xmlDoc at the same time that it will fill a list of questions, where each position i contains a question, with all the attributes we defined previously in the class Question. We can see the pseudo-code of our parse:

```
// var xmlDoc
var question <-xmlDoc.getElementByTagName("question");
var listQuestions <- []
Para i=0 -> question.length faz {
   Escolha(xmlDoc.getElementByTagName("question")[i].getAttribute("type ")){
      case 'toggle':
         //some code
         listQuestions.push(question);
         break;

      case 'radiobutton':
         //some code
         listQuestions.push(question);
         break;

      case 'slider':
         //some code
         listQuestions.push(question);
         break;

      case 'checkbox':
         //some code
         listQuestions.push(question);
         break;
   }

   i <-i+1;
}
```

The code below could quite well be the code used in the "Radiobutton" for creating a question. There are three null values in object creation because this type of button does not have a minimum value, a maximum value, and a scale. In contrast, the "Slider" button would already use these three arguments to construct the question. Whenever we create a question we add it to a list of questions:

```
var question = new
    Question(ask,type,id,answer,id_answer,null,null,null,description);
```

5.5.1 Send answers to server

Once the questionnaire is answered, the user will have to submit the answers. At the time of submission there may be no available Internet connection, however as soon as the application detects an Internet connection it will send the data to the server.

The data sent to the server via the application are: patient_id (optional), questionnaire id, start date, end data, answers and logs. The patient_id is optional since this information only exists when it is a "manager" to responder by "user".

An example of sending answers at the end of the user replying to the questionnaire:

```
  {
       patient_id: "<OPTIONAL>",
       metadata:{
           "quest\_id": ""
           "start\_date": "2017-04-24T13:39:59.123+0100",
           "end\_date": "2017-04-24T13:40:59+0100",
       }
       answers: {
           "2":
       }
       logs: [

       ]
   }
```

In order to be able to send data to the server even when the application is in the background or even closed, it is necessary to use libraries for background job schedule.

React Native Background job (Android)

This library depends on the HeadlessJS of React Native, which is currently only supported on the Android operating system. Registering jobs does not mean that the job is scheduled, registration only tells you that there are jobs to schedule. Jobs do not fire while the application is in the foreground.

Register obj(obj (jobKey,obj job))

Registration must occur globally in the application and not in a life cycle method. Recall that registering the job is not the same as running the job. You need to schedule the job.

Schedule

Schedules a new job. Contrary to register, schedule should only do once.

React Native Background Timer

Through this application it is possible to issue an event periodically even when the application is in the background.

5.6 Logs

The logs of the application are very useful for the most diverse situations: in order to perceive if the user has changed their answers or not throughout the questionnaire; The warnings and / or errors that the application is returning during its operation (for example when it tries to send the responses to the server but there is no Internet network available at the time of sending), etc.

Our logs are divided into the following categories:

- **Analytics (value 0):** All logs of interest from the point of view of analysis of the results. For example, the date and time that the user started responding to the questionnaire as the one that ended. Thus, it is possible to realize how long it takes the patient to respond to the questionnaire.
- **Actions (value 1):** All logs resulting from actions taken by the user, for example whenever he answers a question;
- **Logs (value 2):** All logs of the operation of the application.
- **Warnings (value 3):** Notices of strange events.
- **Error (value 4):** The logs of the errors that the application can return during its operation.

Each row of logs is a JavaScript Object Notation (JSON) object, where it is possible to have null values, with the following information:

- **Category:** Log category.
- **Timestamps:** Log date.
- **Event :** Log event.
- **Type :** Log event type.

- **Value :** Value of the log event.

For example, when the user answers a question, such as "Radiobuttons", the log that we send to the log file in JSON is:

```
(0,new Date().toISOString(),"answer","RadioButtons", id_question + " " +
    id_answers[i])
```

The first value corresponds to its category, as we defined previously; the second value is the date of log submission; the third value is the event, in this case the value is "answer" because it is a response log to a question; the fourth value is the type of event, in this case it is "RadioButtons" since it is the type of event button; and, finally, the value is the set of the ids of the question and the answer. We chose to use the ids to mask the responses and thus ensure more confidentiality of the data.

Chapter 6

Results

Our solution to measure and monitor the quality of life of cancer patients was tested at Instituto Português de Oncologia do Porto (IPO) in a group of aggravated state patients. It was stipulated that these patients would have to answer the questionnaire on the first day and and the following third and sixth month. As the tests only began at the first days of July, we still do not have relevant results to be presented. Besides, due to the aggravated state of the majority of the patients, they may not be able to answer in the following month. In other words, it means that answers related to the third and sixth month of the questionnaire of quality of life might not exist.

However, in this chapter, we present how we analyze the data with a small dataset example. This data are used to test the usability of the solution

The questionnaires used in this dataset have two types of questions: radiobuttons and slider. In the first case the patient has to choose one answer between the given options. In the slider the patient has to answer with the most appropriate level, between a minimum and a maximum value.

6.1 Characterization of the dataset

Our dataset has two different user types. As we can see in the table 6.1, the first one has the questionnaire administered by manager, while the other one has it self-administered. These users tried our solution on two different days.

Table 6.1: Characterization of the dataset.

User	Gender	Age	Pathology	Type of administration
User 1	Male	26	Esophageal	By manager
User 2	Male	28	Esophageal	Self administered

6.2 What are we expecting?

In this sub chapter we present the general process in which a user answers to a question and the expected results.

When an user is answering a question, he goes through a process described in Figure 6.1. There are three important moments to consider: when the question is shown, when he chooses an answer and when he confirms the answer. With these three moments we calculate three different times: time to choose (goes from when the user is shown the question until he chooses the answer), confirmation time (goes from when the user chooses the answer until he confirms it) and the time to answer (sum of the two previous times).

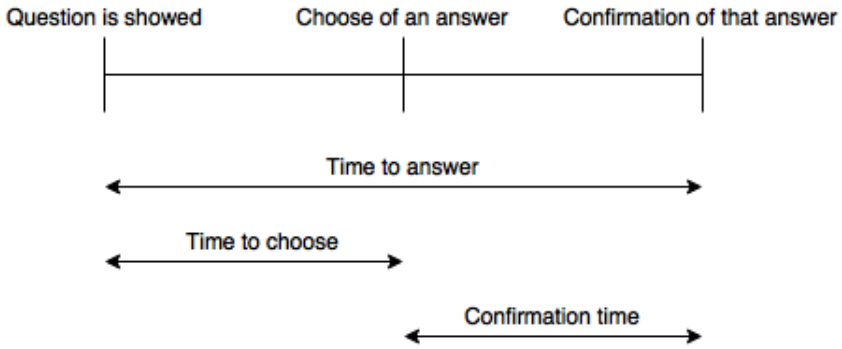

Figure 6.1: Time it takes to answer

We expect the results to vary according to the type of administration (self administration or administration by manager).

- **Questionnaire duration:** We expect the user that answered the questionnaire by himself to have it answered faster than the one who got it administration by manager. We also expect that in the second administration the user takes less time. The average answer time in self administration is expected to be more consistent because contrary to self-administration, when being administered by manager, not only the user but also the manager influences answer times.
- **Choose time:** In administration by manager, the manager reads both the questions and the answers, wait for the user to answer, assimilate the answer and, just then, the answer is chosen. Because of this long process, we expect the average time to choose an answer to be faster in self-administration.
- **Confirmation time:** Like the time to choose, we expect the confirmation time to be faster in self-administration because, for example, in administration by manager, the manager may interfere with the user by asking him if he is sure of his answer.
- **Answers exchanges:** We expect that users change their answers more frequently in self administration because when answering alone they don't feel pressured to answer faster

and tend to be more honest so they get more time to think and possibly change their answers. There are two different types of answer changes: when the time to choose of the first choice is greater than the second, there is a big chance it was just a mistake. On the other hand, if the second one is greater than the first, then the user probably thought better and ultimately changed his answer. In the end, if there are too many mistakes then we probably have to rethink the usability of our solution (eg. buttons, colors, sizes).

6.3 Results

In this sub chapter we present the obtained results with our dataset.

6.3.1 Questionnaire duration

We define by questionnaire duration, the duration a user takes to answer the questionnaire. As we expected, from Figure 6.2, we take that administration by manager takes more time to complete than self administration.

Table 6.2: Questionnaire duration.

Type of administration	Questionnaire duration on the first day	Second Questionnaire duration on the second day	Average answer time on the first questionnaire	Average answer time on the second questionnaire
Manager	0:03:51,2	0:03:44,9	0:00:04,3	0:00:03,9
Self	0:03:42,0	0:02:23,1	0:00:03,7	0:00:02,3

Another important thing we can take from Figure 6.2 (User 1 - Administration by manager; User 2 - Self administration) is that the questionnaire duration decreases from the first to the second administrations. This shows us that the user might have been familiar with the questionnaire during the second administration.

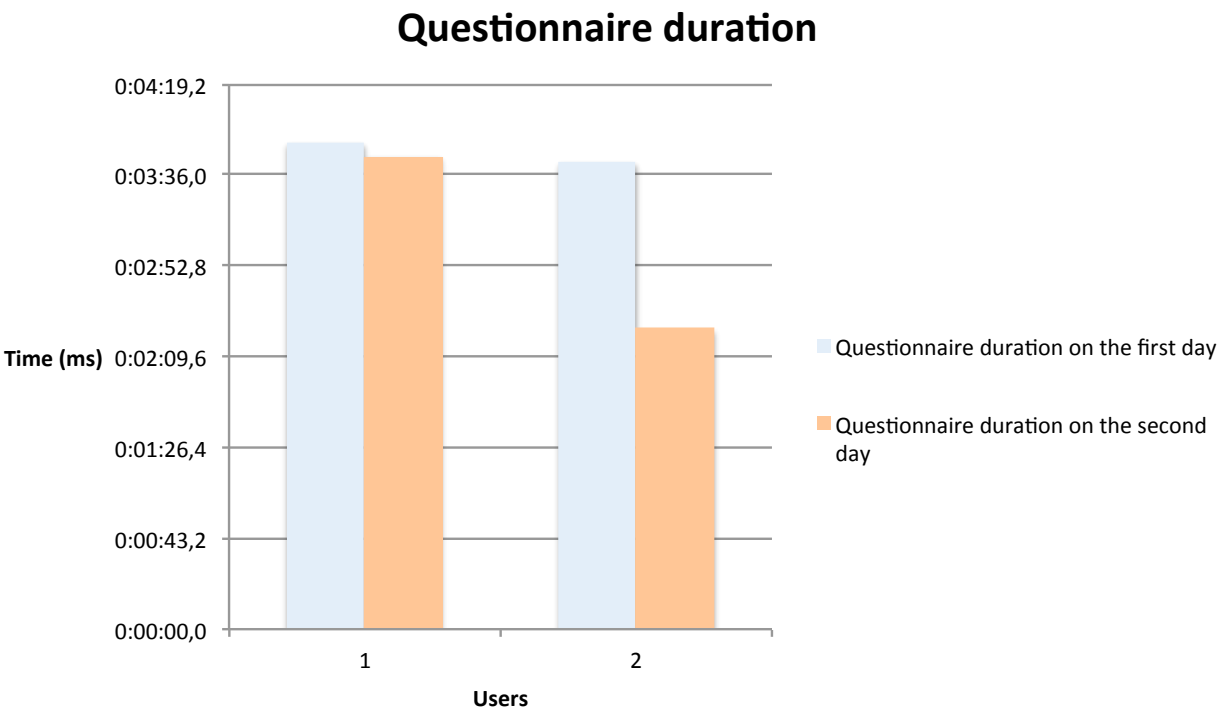

Figure 6.2: Questionnaire duration by type of user.

Average answer time by self administered

As we can see in Figure 6.3, the average answer time is very consistent, even though there two big outliers. The first outlier shown in the graph comes from the second question. This might be caused by the fact that the user wasn't sure of his answer. During the first administration the user took a lot more time than during the second and in the end, he even picked different answers. The second outlier, that comes from the thirtieth question, was possibly caused by a change of answer during the first administration. The time the user took to pick an answer for the first time was inferior to the second pick, that leads us to believe that the user reconsidered his choice.

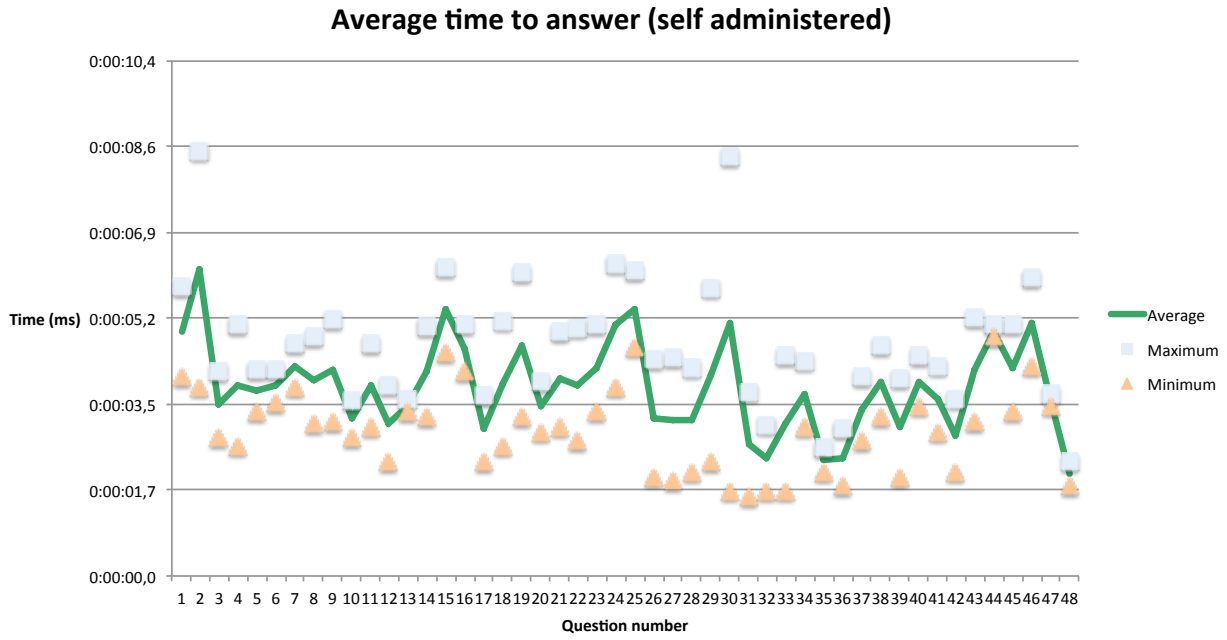

Figure 6.3: Average answer time (self administered)

Average answer time by manager administered

Contrary to self administration, administration by manager has a large spread between average answer times. Even though this happens, there are also moments where the average answer times are more consistent (eg. between questions twenty-eight and thirty-seven), like we can see in Figure 6.4. We believe this is caused by the fact that these question are more objective (eg. "Do you lack appetite?").

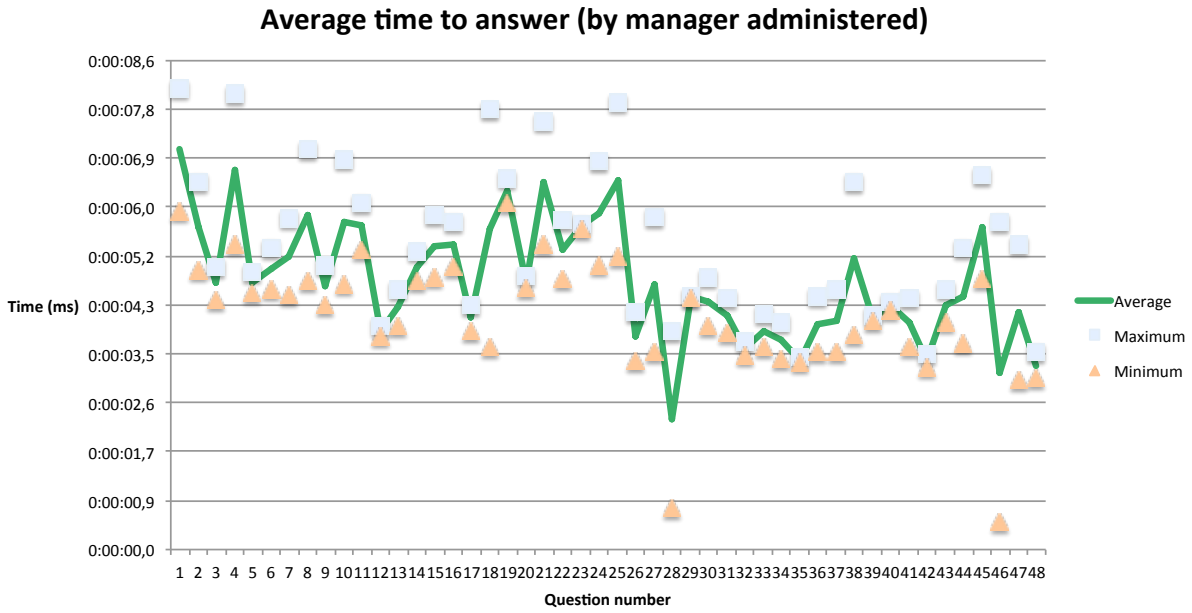

Figure 6.4: Average answer time (by manager administered)

6.3.2 Time to choose

The average answer time varies depending on the type of administration, being the administered by a manager longer than the self-administered one. This happens because having the questionnaire administered by a manager adds some time for the manager to read the questions and answers, and the user to assimilate what the manager just said to select an answer. Figures 6.5 and 6.6 show us the average time the users took to choose their answers.

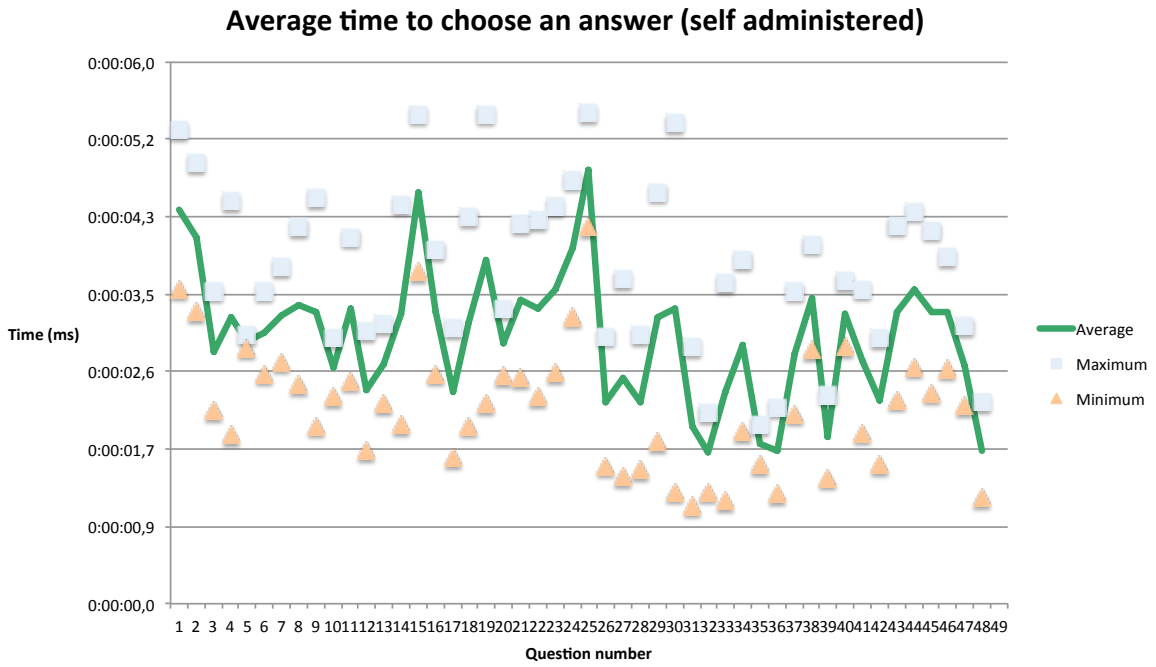

Figure 6.5: Average choose time (self administered)

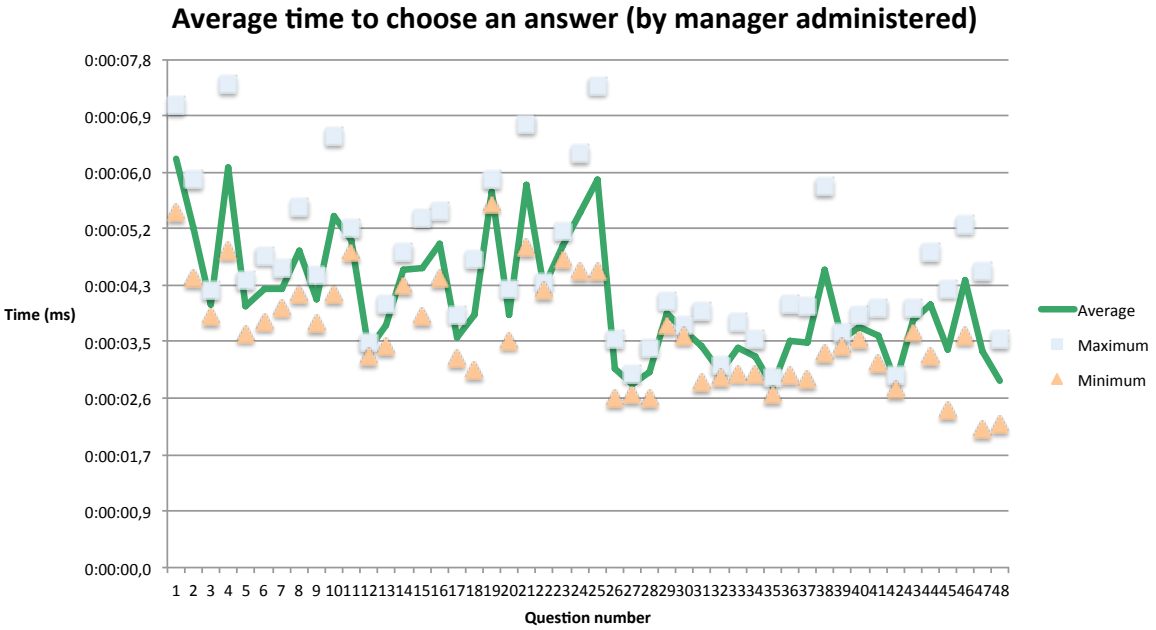

Figure 6.6: Average choose time (by manager administered)

6.3.3 Confirmation time

We expected the average confirmation time to be higher when the questionnaire is administered by manager, however in Figure 6.9, 6.10 and Table 6.6 we see our results show us the opposite.

Table 6.3: Confirmation time.

Type of administration	Confirmation time in questionnaire 1	Confirmation time in questionnaire 2	Average confirmation time
Manager	0:00:00,6	0:00:00,9	0:00:00,7
Self	0:00:00,9	0:00:00,7	0:00:00,8

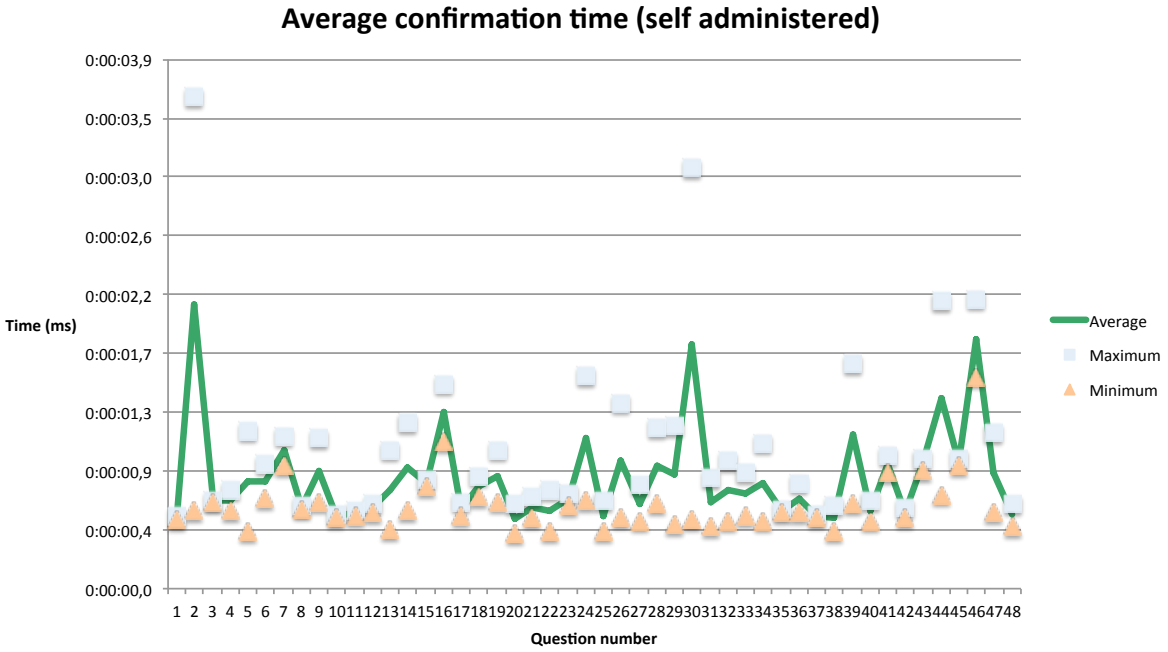

Figure 6.7: Average confirmation time (self administered)

These results have a few outliers in both administering types that might be justified. The high peak outliers shown from the self-administering one might be caused by the user not being using his time exclusively answering the questionnaire, something else could have distracted the user making him take more time to confirm his answer. The administered by manager high peak, however, could have just been the manager asking the user one more time if he was sure of his answer, after noticing some kind of hesitation from the user.

6.3.4 Answer changes

All recorded answer changes were between only one level. Table 6.4 and 6.5 represent what we consider to be two important times to analyze: first answer time, the time it takes for the user to choose his first answer starting from the time the question is shown, and second answer time, the time it takes for the user to choose his second answer starting from the point he chose his

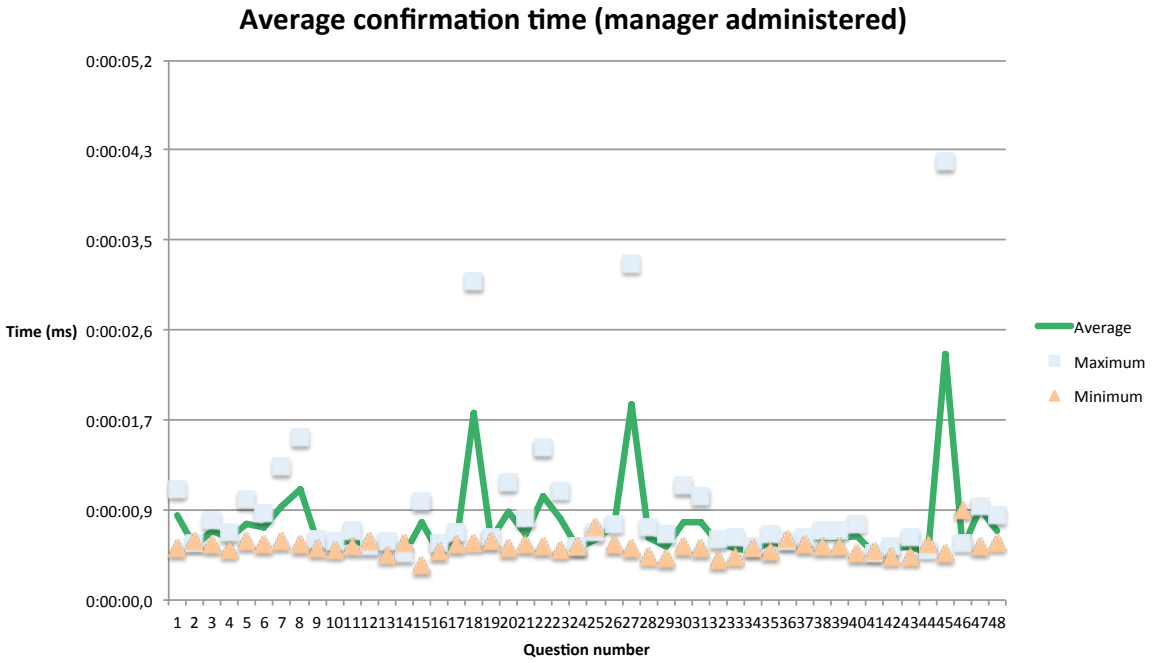

Figure 6.8: Average confirmation time (by manager administered)

first one.

Table 6.4: Answer exchange time (self administered)

Initial time	First answer	Second answer	First answer time	Second answer time
13:47:55,2	13:47:58,0	13:47:58,9	0:00:02,9	0:00:00,9
13:49:44,5	13:49:46,8	13:49:49,9	0:00:02,2	0:00:03,1
13:49:59,7	13:50:02,6	13:50:03,3	0:00:02,9	0:00:00,6
13:50:49,0	13:50:52,6	13:50:53,2	0:00:03,6	0:00:00,6
9:18:49,3	9:18:52,1	9:18:53,0	0:00:02,8	0:00:00,9

Table 6.5: Answer exchange time (administered by manager)

Initial time	First answer	Second answer	First answer time	Second answer time
11:56:20,0	11:56:22,9	11:56:23,4	0:00:03,0	0:00:00,5
8:37:29,9	8:37:35,7	8:37:36,7	0:00:05,8	0:00:01,0
8:38:09,2	8:38:12,2	8:38:12,9	0:00:03,0	0:00:00,7

There were a total of eight changes split between administered by manager and self administered and we believe that all but one were just mistakes since the second answer times are shorter than the first answer times.

6.4 Conclusions

Although our small dataset doesn't allow us to make scalable conclusions, the results were in line with what we expected. The only point our results didn't go along with what we expected was in the confirmation times and even there the times were quite close. We believe this small

difference isn't conclusive.

Since the recorded times are influenced by the user, and even though there is no such thing as a good or bad time, these still offer us good metrics to evaluate the usability of the application.

Chapter 7

Conclusions

After concluding our results met our expectations, in chapter 6 we present a list of completed objectives and future work.

Even though our results could have been more interesting, had we tested the application in Instituto Português de Oncologia do Porto (IPO) sooner (unfortunately that wasn't possible), we still got results that let us draw some important conclusions for future work, like how testing works, what metrics to use and even data analysis.

7.1 Completed objectives

In chapter 1 we defined our objectives for the solution and, at the end, we are able to say that we very satisfied with the results. Our application has the following characteristics and functionalities:

- The application are available and tested in Android.
- Possibility of self-administration or administration by trained and qualified people.
- Patients authentication (through QR Codes).
- XML parser of questionnaires.
- Requests for information to the server.
- Receiving information from the server.

7.2 Future Work

Although, in section 5.3.6 of chapter 5, we told which Application Programming Interface (API) we are going to use to send information to the server when the application closed or running in

the background, unfortunately we weren't able implement this functionality. In addition, we also need to implement notifications for both users and other types of users.

Although the multi-platform development allows to make application available in both operating systems (Android and iOS), we still haven't tested the iOS version. This is one of the next steps we will take.

In addition to this, we also plan to make some minor changes to the user interface to provide an increasingly familiar, intuitive and clear application according to the data we collected from the IPO tests.

Bibliography

[1] OECD. Deaths from cancer (indicator), 2016.

[2] David H. Feeny Guyatt, Gordon H. and Donald L. Patrick. Measuring health-related quality of life. 188.8:622–629, 1993.

[3] Darrell West. How mobile devices are transforming healthcare. *Issues in technology innovation*, 18(1):1–11, 2012.

[4] David Felce and Jonathan Perry. Quality of life: Its definition and measurement. *Research in developmental disabilities*, 16(1):51–74, 1995.

[5] Morag. Farquhar. Definitions of quality of life: a taxonomy. *Journal of advanced nursing*, 22.3:502–508, 1995.

[6] Andrew. Bottomley. The cancer patient and quality of life. 7.2:120–125, 2002.

[7] Kenneth Charles. Calman. Quality of life in cancer patients–an hypothesis. *Journal of medical ethics*, 10.3:124–127, 1981.

[8] Ben-Chieh. Liu. Quality of life indicators in us metropolitan areas. 1976.

[9] Intagliata J. Baker, E. Quality of life in the evaluation of community support systems. 5: 69–79, 1982.

[10] Ira B. Wilson and Paul D. Cleary. Linking clinical variables with health-related quality of life. 273.1:59–65, 1995.

[11] Irene J. Higginson and Alison J. Carr. Using quality of life measures in the clinical setting. 322.7297:1297–1300, 2001.

[12] K Donovan, RW Sanson-Fisher, and S Redman. Measuring quality of life in cancer patients. *Journal of Clinical Oncology*, 7(7):959–968, 1989.

[13] Ronald Feld. Endpoints in cancer clinical trials: is there a need for measuring quality of life? *Supportive Care in Cancer*, 3(1):23–27, 1995.

[14] Jenny Morris, David Perez, and Bronwen McNoe. The use of quality of life data in clinical practice. *Quality of Life Research*, 7(1):85–91, 1997.

[15] G Virone, A Wood, L Selavo, Q Cao, L Fang, T Doan, Z He, and J Stankovic. An advanced wireless sensor network for health monitoring. In *Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, pages 2–4, 2006.

[16] Tia Gao, Christopher Pesto, Leo Selavo, Yin Chen, JeongGil Ko, JongHyun Lim, Andreas Terzis, Andrew Watt, James Jeng, Bor-rong Chen, et al. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 187–192. IEEE, 2008.

[17] JeongGil Ko, Chenyang Lu, Mani B Srivastava, John A Stankovic, Andreas Terzis, and Matt Welsh. Wireless sensor networks for healthcare. *Proceedings of the IEEE*, 98(11):1947–1960, 2010.

[18] Mark A Hanson, Harry C Powell Jr, Adam T Barth, Kyle Ringgenberg, Benton H Calhoun, James H Aylor, and John Lach. Body area sensor networks: Challenges and opportunities. *Computer*, 42(1):58, 2009.

[19] Hande Alemdar and Cem Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688–2710, 2010.

[20] Peter M. Fayers and David Machin. *Quality of life: the assessment, analysis and interpretation of patient-reported outcomes.* 2013.

[21] Miguel Relvas-Silva, Rui Almeida Silva, and Mário Dinis-Ribeiro. Portuguese version of the eortc qlq-oes18 and qlq-og25 for health-related quality of life assessment. *Acta Médica Portuguesa*, 30(1):47–52, 2017.

[22] Neil K. Aaronson, Sam Ahmedzai, Bengt Bergman, Monika Bullinger, Ann Cull, Nicole J. Duez, Antonio Filiberti, Henning Flechtner, Stewart B. Fleishman, Johanna C. J. M. de Haess, Stein Kaasa, Marianne Klee, David Osoba, Darius Razavi, Peter B. Rofe, Simon Schraub, Kommer Sneeuw, Marianne Sullivan, and Fumikazu Takeda. The european organization for research and treatment of cancer qlq-c30: A quality-of-life instrument for use in international clinical trials in oncology. *JNCI: Journal of the National Cancer Institute*, 85(5):365, 1993. doi:10.1093/jnci/85.5.365.

[23] K Bjordal, A De Graeff, PM Fayers, E Hammerlid, Curran van Pottelsberghe, D Curran, Marianne Ahlner-Elmqvist, EJ Maher, JW Meyza, A Bredart, et al. A 12 country field study of the eortc qlq-c30 (version 3.0) and the head and neck cancer specific module (eortc qlq-h&n35) in head and neck patients. *European Journal of Cancer*, 36(14):1796–1807, 2000.

[24] Luis F Oñate-Ocaña, Natalia Velázquez-Monroy, Luis Vázquez, Pablo Espinosa-Mireles-de Villafranca, Elvia Núñez-Rosas, Maira Ovando-Lezama, Diana Vilar-Compte, Gabriela García-Hubard, José F Carrillo, Jane M Blazeby, et al. Clinical validation of the eortc qlq-og25 questionnaire for the evaluation of health-related quality of life in mexican patients with esophagogastric cancers. *Psycho-Oncology*, 21(7):745–753, 2012.

[25] Pernilla Lagergren, Peter Fayers, Thierry Conroy, Hubert J Stein, Orhan Sezer, Richard Hardwick, Eva Hammerlid, Andrew Bottomley, Eric Van Cutsem, Jane M Blazeby, et al. Clinical and psychometric validation of a questionnaire module, the eortc qlq-og25, to assess health-related quality of life in patients with cancer of the oesophagus, the oesophago–gastric junction and the stomach. *European journal of cancer*, 43(14):2066–2073, 2007.

[26] Wei-Chu Chie, Chia-Jih Tsai, Chieh Chiang, and Yung-Chie Lee. Quality of life of patients with oesophageal cancer in taiwan: validation and application of the taiwan chinese (mandarin) version of the eortc qlq-oes18: a brief communication. *Quality of Life Research*, 19(8):1127–1131, 2010.

[27] Kate R Lorig, David S Sobel, Philip L Ritter, Diana Laurent, and Mary Hobbs. Effect of a self-management program on patients with chronic disease. *Effective clinical practice: ECP*, 4(6):256–262, 2000.

[28] Tim Robinson, Thomas Cronin, Haider Ibrahim, Mark Jinks, Timothy Molitor, Joshua Newman, and Jonathan Shapiro. Smartphone use and acceptability among clinical medical students: a questionnaire-based study. *Journal of medical systems*, 37(3):1–7, 2013.

[29] Emilia Bielli, Fabio Carminati, Stella La Capra, Micaela Lina, Cinzia Brunelli, and Marcello Tamburini. A wireless health outcomes monitoring system (whoms): development and field testing with cancer patients using mobile phones. *BMC medical informatics and decision making*, 4(1):1, 2004.

[30] Holly Blake. Innovation in practice: mobile phone technology in patient care. *Br J Community Nurs*, 13(4):160, 2008.

[31] Holly Blake. Mobile phone technology in chronic disease management. *Nursing Standard*, 23(12):43–46, 2008.

[32] Fengqing Zhu, Marc Bosch, Insoo Woo, SungYe Kim, Carol J Boushey, David S Ebert, and Edward J Delp. The use of mobile devices in aiding dietary assessment and evaluation. *IEEE Journal of Selected Topics in Signal Processing*, 4(4):756–766, 2010.

[33] Deborah Estrin and Ida Sim. Open mhealth architecture: an engine for health care innovation. *Science*, 330(6005):759–760, 2010.

[34] Donna S Eng and Joyce M Lee. The promise and peril of mobile health applications for diabetes and endocrinology. *Pediatric diabetes*, 14(4):231–238, 2013.

[35] Robert Hurling, Michael Catt, Marco De Boni, Bruce Fairley, Tina Hurst, Peter Murray, Alannah Richardson, and Jaspreet Sodhi. Using internet and mobile phone technology to deliver an automated physical activity program: randomized controlled trial. *Journal of medical Internet research*, 9(2):e7, 2007.

[36] Sonia A Lamel, Kristin M Haldeman, Haines Ely, Carrie L Kovarik, Hon Pak, and April W Armstrong. Application of mobile teledermatology for skin cancer screening. *Journal of the American Academy of Dermatology*, 67(4):576–581, 2012.

[37] Andre Charland and Brian LeRoux. Mobile application development: Web vs. native. *Queue*, 9(4):20:20–20:28, April 2011. ISSN: 1542-7730. doi:10.1145/1966989.1968203.

[38] Niclas Hansson and Tomas Vidhall. Effects on performance and usability for cross-platform application development using react native. Master's thesis, Linköping University, Human-Centered systems, 2016.

[39] Henning Heitkötter, Sebastian Hanschke, and Tim A Majchrzak. Evaluating cross-platform development approaches for mobile applications. In *International Conference on Web Information Systems and Technologies*, pages 120–138. Springer, 2012.

[40] Oscar Axelsson and Fredrik Carlström. Evaluation targeting react native in comparison to native mobile development, 2016. Student Paper.

[41] CP Rahul Raj and Seshu Babu Tolety. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In *India Conference (INDICON), 2012 Annual IEEE*, pages 625–629. IEEE, 2012.

[42] Isabelle Dalmasso, Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. Survey, comparison and evaluation of cross platform mobile application development tools. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 323–328. IEEE, 2013.

[43] Aymen Beshir. Cross-platform development with react native. (16053):32, 2016.

[44] Artemij Fedosejev. *React.Js Essentials*. Packt Publishing, 2016. ISBN: 1783551623, 9781783551620.

[45] Ethan Holmes and Tom Bray. *Getting Started with React Native*. Packt Publishing Ltd, 2015.

Appendix A

Attachments

EORTC QLQ – OG25

Por vezes, os pacientes reportam a ocorrência dos seguintes sintomas ou problemas. Por favor, indique em que medida sentiu estes sintomas ou problemas durante a última semana. Responda fazendo um círculo à volta do número que melhor se adequa à sua situação.

Durante a última semana:		**Não**	**Um pouco**	**Bastante**	**Muito**
31.	Teve problemas ao ingerir alimentos sólidos?	1	2	3	4
32.	Teve problemas ao ingerir alimentos liquidificados ou moles?	1	2	3	4
33.	Teve problemas ao ingerir líquidos?	1	2	3	4
34.	Teve dificuldade em usufruir (ter prazer) das suas refeições?	1	2	3	4
35.	Sentiu-se cheio(a) pouco depois de começar a comer?	1	2	3	4
36.	Demorou muito tempo para terminar as refeições?	1	2	3	4
37.	Teve dificuldade em comer?	1	2	3	4
38.	Teve indigestão ácida ou ardor?	1	2	3	4
39.	Sentiu ácido ou bílis na boca?	1	2	3	4
40.	Teve mal-estar ao comer?	1	2	3	4
41.	Teve dores ao comer?	1	2	3	4
42.	Teve dores na zona do estômago?	1	2	3	4
43.	Teve mal-estar na zona do estômago?	1	2	3	4
44.	Tem pensado na sua doença?	1	2	3	4
45.	Tem-se preocupado com a sua saúde no futuro?	1	2	3	4
46.	Teve problemas em comer diante de outras pessoas?	1	2	3	4
47.	Sentiu a boca seca?	1	2	3	4
48.	Teve problemas com o seu paladar?	1	2	3	4
49.	Sentiu-se menos atraente fisicamente em resultado da sua doença ou do tratamento?	1	2	3	4

Avance para a próxima página

Durante a última semana:		**Não**	**Um pouco**	**Bastante**	**Muito**
50.	Teve dificuldade em engolir a sua saliva?	1	2	3	4
51.	Engasgou-se ao engolir?	1	2	3	4
52.	Tossiu?	1	2	3	4
53.	Teve dificuldade em falar?	1	2	3	4
54.	Tem-se preocupado por ter um peso demasiado baixo?	1	2	3	4
55.	Responda a esta pergunta apenas se registou queda de cabelo: Se assim foi, sentiu-se incomodado(a) com a perda de cabelo?	1	2	3	4

Appendix B

Attachments

EORTC QLQ – OES18

Às vezes os doentes relatam que têm os seguintes sintomas ou problemas. Por favor, indique em que medida sentiu estes sintomas ou problemas durante a última semana. Por favor, responda fazendo um círculo ao redor do número mais adequado ao seu caso.

Durante a última semana:	**Não**	**Um pouco**	**Bas-tante**	**Muito**
31. Você conseguiu comer alimentos sólidos?	1	2	3	4
32. Você conseguiu comer alimentos liquidificados ou moles?	1	2	3	4
33. Você conseguiu beber líquidos?	1	2	3	4
34. Você tem tido dificuldade para engolir a saliva?	1	2	3	4
35. Ao engolir, você tem-se engasgado?	1	2	3	4
36. Você tem tido dificuldade em ter prazer nas suas refeições?	1	2	3	4
37. Você tem-se sentido cheio(a) muito rapidamente?	1	2	3	4
38. Você tem tido dificuldade ao comer?	1	2	3	4
39. Você tem tido problemas em comer na presença de outras pessoas?	1	2	3	4
40. Você tem sentido a sua boca seca?	1	2	3	4
41. Os alimentos e as bebidas têm tido um sabor diferente do usual?	1	2	3	4
42 Você tem tido problemas ao tossir?	1	2	3	4
43. Você tem tido problemas ao falar?	1	2	3	4
44. Você tem sentido indigestão com acidez ou queimor?	1	2	3	4
45. Você tem sentido problemas de gosto amargo ou acidez que vêm à boca?	1	2	3	4
46. Você tem tido dor enquanto você come?	1	2	3	4
47. Você tem tido dor no peito?	1	2	3	4
48. Você tem tido dor no estômago?	1	2	3	4

Appendix C

Attachments

EORTC QLQ-C30 (version 3)

Gostaríamos de conhecer alguns pormenores sobre si e a sua saúde. Responda você mesmo/a, por favor, a todas as perguntas fazendo um círculo à volta do número que melhor se aplica ao seu caso. Não há respostas certas nem erradas. A informação fornecida é estritamente confidencial.

Escreva as iniciais do seu nome: ⌊ ⌊ ⌊ ⌊ ⌋

A data de nascimento (dia, mês, ano): ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌋

A data de hoje (dia, mês, ano): 31 ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌊ ⌋

		Não	Um pouco	Bastante	Muito
1.	Custa-Ihe fazer esforços mais violentos, por exemplo, carregar um saco de compras pesado ou uma mala?	1	2	3	4
2.	Custa-lhe percorrer uma grande distância a pé?	1	2	3	4
3.	Custa-lhe dar um pequeno passeio a pé, fora de casa?	1	2	3	4
4.	Precisa de ficar na cama ou numa cadeira durante o dia?	1	2	3	4
5.	Precisa que o/a ajudem a comer, a vestir-se, a lavar-se ou a ir à casa de banho?	1	2	3	4

Durante a última semana :

		Não	Um pouco	Bastante	Muito
6.	Sentiu-se limitado/a no seu emprego ou no desempenho das suas actividades diárias?	1	2	3	4
7.	Sentiu-se limitado/a na ocupação habitual dos seus tempos livres ou noutras actividades de laser?	1	2	3	4
8.	Teve falta de ar?	1	2	3	4
9.	Teve dores?	1	2	3	4
10.	Precisou de descansar?	1	2	3	4
11.	Teve dificuldade em dormir?	1	2	3	4
12.	Sentiu-se fraco/a?	1	2	3	4
13.	Teve falta de apetite?	1	2	3	4
14.	Teve enjoos?	1	2	3	4
15.	Vomitou?	1	2	3	4

Por favor, passe à página seguinte

Durante a última semana :	**Não**	**Um pouco**	**Bastante**	**Muito**
16. Teve prisão de ventre?	1	2	3	4
17. Teve diarreia?	1	2	3	4
18. Sentiu-se cansado/a?	1	2	3	4
19. As dores perturbaram as suas actividades diárias?	1	2	3	4
20. Teve dificuldade em concentrar-se, por exemplo, para ler o jornal ou ver televisão?	1	2	3	4
21. Sentiu-se tenso/a?	1	2	3	4
22. Teve preocupações?	1	2	3	4
23. Sentiu-se irritável?	1	2	3	4
24. Sentiu-se deprimido/a?	1	2	3	4
25. Teve dificuldade em lembrar-se das coisas?	1	2	3	4
26. O seu estado físico ou tratamento médico interferiram na sua vida familiar?	1	2	3	4
27. O seu estado físico ou tratamento médico interferiram na sua actividade social?	1	2	3	4
28. O seu estado físico ou tratamento médico causaram-Ihe problemas de ordem financeira?	1	2	3	4

Nas perguntas que se seguem faça um círculo à volta do número, entre 1 e 7, que melhor se aplica ao seu caso

29. Como classificaria a sua saúde em geral durante a última semana?

1	2	3	4	5	6	7
Péssima						Óptima

30. Como classificaria a sua qualidade de vida global durante a última semana?

1	2	3	4	5	6	7
Péssima						Óptima